Titre: Title:	High Quality Delay Testing Scheme for a Self-Timed Microprocessor
Auteur: Author:	Omar Al-Terkawi Hasib
Date:	2019
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Al-Terkawi Hasib, O. (2019). High Quality Delay Testing Scheme for a Self-Timed Microprocessor [Thèse de doctorat, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/4011/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

High quality delay testing scheme for a self-timed microprocessor

OMAR AL-TERKAWI HASIB
Département de génie électrique

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie Électrique

Août 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

High quality delay testing scheme for a self-timed microprocessor

présentée par **Omar AL-TERKAWI HASIB**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Yves AUDET, président
Yvon SAVARIA, membre et directeur de recherche
Claude THIBEAULT, membre et codirecteur de recherche
Yves BLAQUIÈRE, membre
Benoit NADEAU-DOSTIE, membre externe

DEDICATION

To my mother,
without you this would not have been possible...

ACKNOWLEDGEMENTS

During these five years of my PhD, I have worked with many wonderful and intelligent people, and was surrounded by lots of love and support that made this journey bearable. I would like to start this section by giving thanks to God for facilitating this work and providing me with the means, time, health and capabilities to absorb all this knowledge, and for the motivation to continue this work to the end. Next, I would like to thank my family for being supportive and understanding during this long journey of higher education. I realize that I could have been much more helpful to them if I was working and making 3 times the money. So I appreciate their patience with me. Of course, I could not have made it this far if it wasn't for my mother, who supported me and encouraged me (in many different ways) to finish as soon as possible. I love you Mom!

I would also like to thank both of my supervisors, Yvon Savaria and Claude Thibeault. Yvon have encouraged me over the years, though I was reluctant, to continue my graduate studies and gave me a good financial incentive to go back and do this PhD. Claude was very generous in both knowledge and financial support. Delay testing is a hugely new domain to me and Claude's experience have helped me present my work in the most relevant way to the people in this research domain. I also must thank both my supervisors Yvon Savaria and Claude Thibeault for all the guidance and support that they provided me during these past five years. Not to mention, their effort into accommodating my need to finish this dissertation in time, and their patience in reviewing my papers and correcting my endless typos.

This journey would have been much more difficult if it wasn't for the great people that were around me. I would like to thank my desk neighbor and friend Mickael Fiorentino for listening to all my ranting about the research community and the difficulties of PhD, and for making time to discuss with me the different aspects of my work and idealistic ideas of how the world should be. I would like to also thank my other lab mates Michel Gemieux, Safa Berrima, and recently Erika Miller, for being great lab mates and putting up with all the noise that I made arguing with Mickael.

This work would not have been possible if it was not for the awesome help of my friend and great system admin Rejean Lepage. He put in extremely great effort to setup the proper environment and security measures to install the latest technologies on our servers. He also would cut time from his vacation to remote login and fix issues in the system so that our

work would not be blocked. Thank you Rejean.

Being thousands of miles away from home meant that loneliness would be a common visitor for me. However, the great company of my friends in Montreal made it always much easier. I would like to thank first Ahmad Haidar and Mohammad Fathy. I cannot put in words how irreplaceable these friends are. I would also like to thank Faycal Mounaim, Maraym Tabtabai and Mahya Dehbozorgi for the endless fun we had during these years with the rest of the group and especially for inviting me to their houses and sharing heart-warming home made meals. I also shouldn't forget my online friends from back home. You guys on twitter have heard me rant more than anyone I know and still did not unfollow me. You also made me feel as if I never left home. Thank you.

Finally, I would like to thank CMC Microsystems and CMP for providing the CAD tools and access to the 28nm PDK, Octasic for providing financial support and scientific guidance, and the Natural Sciences and Engineering Research Council of Canada for providing partial funding. From Octasic, I would like to specifically thank Daniel Crepeau for taking the time to help me make the test in this work functional. Lastly, many thanks goes to CMC and professor Gordon Roberts (from McGill university) for the access to the Teradyne FLEX tester.

RÉSUMÉ

La popularité d'internet et la quantité toujours croissante de données qui transitent à travers ses terminaux nécessite d'importantes infrastructures de serveurs qui consomment énormément d'énergie. Par conséquent, et puisqu'une augmentation de la consommation d'énergie se traduit par une augmentation des coûts, la demande pour des processeurs efficaces en énergie est en forte hausse. Une manière d'augmenter l'efficacité énergétique des processeurs consiste à moduler la fréquence d'opération du système en fonction de la charge de travail. Les processeurs endochrones et asynchrones sont une des solutions mettant en œuvre ce principe de modulation de l'activité à la demande. Cependant, les méthodes de conception non conventionnelles qui leur sont associées, en particulier en termes de testabilité et d'automation, sont un frein au développement de ce type de systèmes.

Ce travail s'intéresse au développement d'une méthode de test de haute qualité adressée aux pannes de retards dans une architecture de processeur endochrone spécifique, appelée AnARM. La méthode proposée consiste à détecter les pannes à faibles retards (PFR) dans l'AnARM en tirant profit des lignes à délais configurables intégrées. Ces pannes sont connues pour passer au travers des modèles de pannes de retards utilisés habituellement (les pannes de retards de portes). Ce travail s'intéresse principalement aux PFR qui échappent à la détection des pannes de retards de portes mais qui sont suffisamment longues pour provoquer des erreurs dans des conditions normales d'opération. D'autre part, la détection de pannes à très faibles retards est évitée, autant que possible, afin de limiter le nombre de faux positifs. Pour réaliser un test de haute qualité, ce travail propose, dans un premier temps, une métrique de test dédiée aux PFR, qui est mieux adaptée aux circuits endochrones, puis, dans un second temps, une méthode de test des pannes de retards basée sur la modulation de la vitesse des lignes à délais intégrés, qui s'adapte à un jeu de vecteurs de test préexistant.

Ce travail présente une métrique de test ciblant les PFR, appelée pourcentage de marges pondérées (PoMP), ainsi qu'un nouveau modèle de test pour les PFR (appelé test de PFR idéal). La métrique PoMP est construite de manière flexible, ce qui lui permet de s'adapter aux informations du circuit et de l'environnement de test disponibles. Ainsi, comme l'AnARM utilise plusieurs vitesses d'opérations pendant le test, le calcul de la PoMP est particulièrement sensible aux effets que la modulation de fréquence peut induire sur la qualité du test. En particulier, des mesures spéciales sont prises pour éviter le sur-test du circuit. Le sur-test peut mener à la conclusion que le circuit à tester est défectueux, à cause de pannes

de retard qui sont en réalité trop courtes pour affecter ses fonctionnalités. Une infrastructure logicielle a été mise au point pour le calcul de la PoMP, permettant de la comparer avec d'autres métriques de la littérature sur de larges échantillons procédés-tensions-températures. Les simulations sont réalisées sur une sélection de circuits de références synthétisés en technologie 28 nm FD-SOI de STMicroelectronics. Les résultats montrent que la PoMP est la métrique la plus sensible aux changements qui affectent le test et le sur-test des PFR.

Ce travail propose une méthode d'optimisation du test des PFR basée sur la PoMP. La méthode associe des fréquences d'horloges de test avec des vecteurs de test préexistant pour maximiser la qualité du test. La méthode employée consiste, dans un premier temps, à classifier les meilleurs choix de test pour chaque panne dans le circuit, puis, dans un second temps, à grouper les pannes qui partagent les mêmes conditions de fonctionnement (i.e, fréquence d'horloges de test, vecteurs de tests, et masquage). Les techniques de groupement et de masquage permettent de significativement diminuer le nombre de vecteurs de test tout en conservant une bonne qualité de test des PFR. Sur une sélection de circuits de références, la méthode proposée est capable d'améliorer la valeur de la métrique PoMP jusqu'à 32.97% comparé à un test à vitesse nominale. Le test optimisé des PFR qui est proposé, ainsi que les tests à vitesse nominale développés précédemment, sont mis en œuvre sur les 25 puces AnARM fabriquées en technologie 28 nm FD-SOI de STMicroelectronics. L'AnARM inclut quatre fréquences d'horloges pour le test : une à vitesse nominale, et trois à vitesses supérieures. Comparé au test à vitesse nominale, l'optimisation du test des PFR améliore la qualité du test de l'AnARM, mesuré par la PoMP, de 60.03% à 69.71% en utilisant les lignes à délais intégrés, pour des vecteurs de test identiques. De plus, le nombre de vecteurs de test final n'augmente que d'un facteur 4 par rapport au test à vitesse nominale. Si la vitesse des lignes à délais variables est ajustée, la méthode d'optimisation peut atteindre une qualité de test de 89.01%. Les résultats post-fabrication obtenus par le test à vitesse nominale et le test optimisé des PFR sur les 25 puces de l'AnARM qui sont présentés dans ce travail, confirment l'efficacité de la méthode de test des PFR proposée.

ABSTRACT

The popularity of the Internet and the huge amount of data that is transfered between devices nowadays requires very powerful servers that demand lots of power. Since higher power consumptions mean more expenses to companies, there is an increase in demand for power efficient processors. One of the ways to increase the power efficiency of processors is to adapt the processing speeds and chip activity according the needed computation load. Self-timed or asynchronous processors are one of the solutions that apply this principle of activity on demand. However, their unconventional design methodology introduces several challenges in terms of testability and design automation.

This work focuses on developing a high quality delay test for a specific architecture of self-timed processors called the AnARM. The proposed delay test focuses on catching effective small-delay defects (SDDs) in the AnARM by taking advantage of built-in configurable delay lines. Those defects are known to escape one of the most commonly used delay fault models (the transition delay fault model). This work mainly focuses on effective SDDs which can escape transition delay fault testing and are large enough to fail the circuit under normal operating conditions. At the same time, catching very small delay defects is avoided, when possible, to avoid falsely failing functional chips. To build the high quality delay test, this work develops an SDD test quality metric that is better suited for circuits with adaptable speeds. Then, it builds a delay test optimizer that adapts the built-in delay lines speeds to a preexisting at-speed pattern set to create a high quality SDD test.

This work presents a novel SDD test quality metric called the weighted slack percentage (WeSPer), along with a new SDD testing model (named the ideal SDD test model). WeSPer is built to be a flexible metric capable of adapting to the availability of information about the circuit under test and the test environment. Since the AnARM can use multiple test speeds, WeSPer computation takes special care of assessing the effects of test frequency changes on the test quality. Specifically, special care is taken into avoiding overtesting the circuit. Overtesting will cause circuits under test to fail due to defects that are too small to affect the functionality of these circuits in their present state. A computation framework is built to compute WeSPer and compare it with other existing metrics in the literature over a large sets of process-voltage-temperature computation points. Simulations are done on a selected set of known benchmark circuits synthesized in the 28nm FD-SOI technology from STMicroelectronics. The results show that WeSPer is the most accurately sensitive metric

to changes that effect SDD test escapes and overtesting.

An SDD test optimization method based on WeSPer is proposed in this work. The method pairs test clock speeds with preexisting test patterns to maximize the delay test quality. This is done by listing the best choices to test each fault in the circuit, then grouping faults that share the same test conditions (i.e. test clock speed, patterns and masking). The grouping and masking techniques ensure that the final pattern count decreases without a significant decrease in the SDD test quality. The proposed SDD test optimization method can be applied to the AnARM structure, as well as synchronous systems that are compatible with faster-than-at-speed testing. On a set of benchmark circuits, the proposed technique is able to improve the WeSPer value by up to 32.97% (a relative increase of 60.98%) compared to classical at-speed testing using the same pattern set. The proposed optimized SDD test, along with a previously developed at-speed test, are applied on 25 AnARM chips fabricated in the 28nm FD-SOI technology from STMicroelectronics. The AnARM includes 4 levels of test clocks speeds: one at-speed and 3 faster-than-at-speed. Compared to at-speed testing, the optimization SDD test improved the SDD test quality of the AnARM, measured by WeSPer, from 60.03% to 69.71% using the built-in delay lines with the same at-speed test pattern set. Moreover, the final pattern count grows by no more than 4 times the initial at-speed test. If the built-in delay line speed are adjusted, the optimization technique is able to further increase the test quality to 89.01%. The post-silicon at-speed test results and post-silicon optimized SDD test for 25 AnARM chips are presented in this work. The results validate the effectiveness of the proposed SDD test.

Finally it is should be noted that, although the goal of this work is to develop a high quality delay test for the AnARM structure, the proposed metric (WeSPer) along with the proposed SDD test optimization method are not limited to this specific structure. Both WeSPer, and the proposed optimization method for SDD testing, can be applied to classical synchronous systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

ATE	Automatic test equipment
ATPG	Automatic test pattern generation
CAD	Computer-Aided Design
CDL	Configurable delay line
CLC	Configurable logic cloud
CL	confidence level
CUT	Circuit under test
DPM	Defect per million
DTC	Delay Test Coverage
DUT	Device under test
EU	Execution unit
FAST	Faster-than-at-speed testing
JSON	JavaScript Object Notation
LOC	launch-on-capture
LOS	launch-on-shift
MSD	Mean slack difference
PCB	Printed circuit board
PDF	Path delay fault
PDK	Process design kit
PLL	Phase-locked loop
PVT	Process-voltage-temperature
SDDC^{Q}	Quadratic Small-Delay Defect Coverage
SDDC	Small-Delay Defect Coverage
SDD	Small-Delay Defect
SDFC	Statistical Delay Fault Coverage
SDQL	Statistical Delay Quality Level
SDQM	Statistical Delay Quality Model
SEDD	Smallest effective delay defect
STA	Static Timing Analysis
TDFC	Transition delay fault coverage
TDF	Transition delay fault
TOPer	Total Overtesting Percentage
WeSPer	Weighted Slack Percentage

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 Overview and Motivation

In this era of mobile devices and very powerful servers that run the Internet, designs with high power efficiency are in demand. Classical synchronous systems that waste power on clock trees and constant circuit activity are an unattractive solution, from a power efficiency point of view, when compared to event-triggered asynchronous design methodologies. Synchronous designs are however, faster than standard asynchronous systems. This is due to the fact that most asynchronous designs waste time on managing the timing of operations (e.g. hand-shaking), and thus tend to be slower than their synchronous counterparts. One of the unique design methods which does not suffer from this problem is the self-timed (source-synchronous) design presented by Octasic for their digital signal processor (DSP) chip [1]. Their DSP design shows a comparable system performance in terms of speed, while having advantages in power efficiency and area consumption. It has been shown that clocks and their distribution networks can consume up to 40% of the total chip power [2]. Removing synchronous clocks to design an asynchronous system would definitely reduce the power consumption, but the unconventional design methodologies of such systems introduce several challenges in terms of design, testability and design automation with industrial tools [3]. Standard computer-aided design (CAD) softwares cannot be fully used to properly synthesis and test unconventional asynchronous designs without putting a great deal of effort into adapting them.

This work is a part of a bigger project that aims to design a highly testable self-timed microprocessor, called *AnARM*, which uses the Octasic self-timed architecture. This work targets specifically the development of high quality delay test methods to test small-delay defects (SDDs) taking advantage of the unique Octasic self-timed clocking structure. In this chapter, a general introduction to delay testing and the Octasic structure will be presented before further discussing the challenges and goals of this work.

1.2 Delay Testing

Classical synchronous circuits are the base for most modern processors that are embedded in almost every smart device on the market. The correct operation of synchronous circuits relies on a timing relation between the processed data and the operating clock. Fig 1.1 shows a simplified example of a synchronous circuit. In order for a synchronous circuit to operate

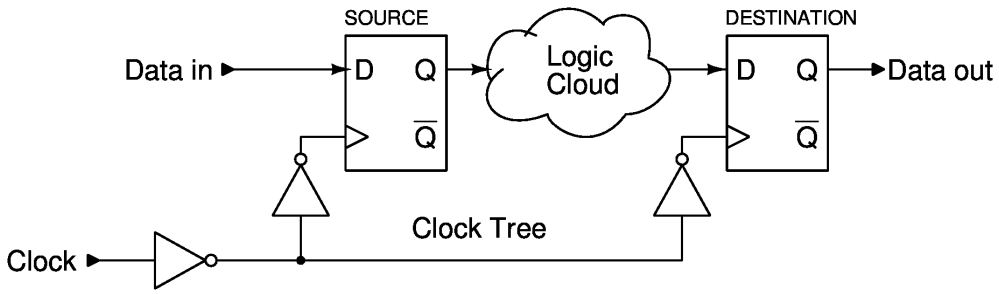

Figure 1.1 Synchronous circuit example.

correctly, there are two main timing constraints that need to be satisfied: the setup timing constraint, and the hold timing constraint [4]. These two constraints govern the delay of the logical cloud and clock network delays with respect to the operating clock period and flip-flop stability time requirements (setup time and hold time). With respect to the logic cloud delay, the setup constraint for circuit circuits can be written as:

$$t_{logic} \leq T_{sys} - t_{setup} - t_{CK-Q} - t_{skew} \tag{1.1}$$

Whereas the hold constraint can be written as:

$$t_{logic} \geq t_{hold} - t_{CK-Q} + t_{skew} \tag{1.2}$$

Where t_{logic} is the combinational logic cloud delay, T_{sys} is the operating (system) clock period, t_{CK-Q} is the delay for the data to appear at the output of the flip-flop after the clock edge arrival, t_{setup} is the setup time for the output flip-flops, t_{hold} is the hold time for the output flip-flops and t_{skew} is the clock tree timing skew.

All of the delays in equations (1.1) and (1.2), except the external clock period, are susceptible to delay variation in CMOS circuits. There are many sources of delay variations, such as process variation, temperature variation, power supply variations, cross-talk and aging [5–8]. To account for most of these variations during the design stage, a static timing analysis (STA) tool is used to find the worst delay in a circuit in order to determine the operating frequency. After fabrication, the speed of the same circuit will vary due to process variations, and sometimes as a consequence, chips that contain no detected fault and that meet a minimum level of performance may be binned into speed categories. Speed variations in fabricated chips are expected and are modeled in the libraries that are provided from the foundry.

However, recent CMOS technologies are becoming extremely small and the fabrication process is becoming more complicated and more defect prone than before [9,10]. This brings an increase to the occurrence rate of defects that may not be catastrophic but that add undesirable delays to the circuit. Delay defects can be caused for example by weak connections (resistive short/opens) which lead to unexpected changes in the delays of a circuit [5, 11]. These added delays cannot be predicted from the design stages and can be difficult to find with normal functional testing. Information on delay defects are not normally included in the standard process design kit (PDK).

The main reason for applying delay testing is to catch these unexpected delay increases that can happen due to inaccuracies in the fabrication process. Delay testing checks thoroughly that the timing constraints are not violated, and that the chip can operate at the rated speed. This is done by carefully selecting the input data, applying a test clock at a certain speed and comparing the results with the expect ones. With millions of transistors packed into modern high speed chips, the challenge in delay testing is to test the entire chip in a reasonable time without missing any defect that can cause the chip to fail in the field.

There are four main aspects that define a delay test: the delay fault model, the test clock speed, the test structure and the applied test pattern. The choice of delay fault model defines how the applied pattern will be generated, whereas the test clock speed defines test structures that need to be inserted into the circuit under test (CUT) and the requirements of the tester. We will introduce here briefly the delay fault models used in testing, how to apply an at-speed delay test using scan based testing, and discuss SDDs. As this work does not contribute to test pattern generation algorithms, we will not delve into the specifics of automatic test pattern generation (ATPG) for delay testing.

1.2.1 Delay Fault Models

A fault is an abstract representation of the defect on a functional level. Delay faults in this case are represented as added delays to the circuit. The two main categories of delay fault models, that are well known in the literature, are the path delay fault (PDF) model [12] and the transition delay fault (TDF) model [13].

In the PDF model, the defect is assumed to be distributed along the path of the signal, and a fault is detected when the total delay along the path exceeds the testing clock period. The PDF model defines two types of faults per path, one for a falling transition and one for

a rising transition. This means that the numbers of delay faults in a given circuit is twice the number of paths in it. In addition, paths under the PDF model are categorized based on the ability to measure the delay of the path independently from other paths in the CUT [14]. The testable paths in PDF are generally categorized into *robust*, *strong non-robust*, *weak non-robust* and *functional*. This categorization is based on the ability to sensitize a path in a given way using the inputs. The conditions for proper sensitization of paths in PDF are very difficult to meet and so its application is limited to a small set of paths (normally critical paths) [14–17]. Moreover, the number of paths in a circuit can grow exponentially with the number of logic gates [18] making it difficult, if not impossible, to test all paths in a reasonable time. Nonetheless, the PDF based delay test is a very thorough test and is often used to check the critical paths of a circuit.

The other widely known model is the TDF model. The TDF model assumes a single defect on each node of a circuit. The defect (if it exists) is assumed to be big enough that it can be detected regardless of the slack of the sensitized path. This approximation makes the TDF model a much simpler model, and allows for the reuse of the ATPG from the common stuck-at-fault testing. In the stuck-at-fault model, a node is assumed to be stuck at a single value (0 or 1), which corresponds in a way to the two fault types that are defined under the TDF model, *slow-to-rise* or *slow-to-fall*. In contrast to the PDF model, in the TDF model, the number of faults to be tested grows linearly with the number of logic gates in the circuit [18]. The simplicity of the TDF model comes with a disadvantage that allows for some delay defects to remain undetected. The next subsection will elaborate on this point. Note that this work is built around the TDF model and will try to address some of those test escapes.

As we will indicate in the literature review, there are some attempts to create a hybrid model of the two delay fault models, as in [19]. However, this work is built on the TDF model due to its simplicity, popularity and compatibly with the existing test tools. Thus, from this point onward the text will focus on the TDF model.

1.2.2 Small Delay Defects: Definitions and Challenges

As mentioned previously, in the TDF model, the assumption is that the delay defect is large enough to be detected on a node regardless of the path used for testing. If the defect was not large enough, then it would escape the test undetected. The question here is whether catching such a small defect critical? The answer to the question depends on how small is

small, and how reliable the chip needs to be. Some defects can be large enough to fail the circuit under normal operation, yet small enough to pass a TDF based test. Others can be small enough to cause no observed failure today, but might grow in the near future to be a problem. Others can be so small that it would be difficult to distinguish them from delay variation induced by the process, cross-talk or supply noise. To better understand the work presented here, we will start by defining all the terms that will be frequently used in relation to delay defects and delay testing.

- ***Normal operating conditions***: this refers to the system clock speed, temperature and supply voltage under which the CUT is expected to operate.
- ***Gross delay defect***: A delay defect that will always be caught in a traditional TDF delay test if tested under normal operating conditions.
- ***Small-delay defect (SDD)*** : a delay defect that adds a small amount of delay to a path in a circuit and that can potentially escape traditional TDF delay testing and remain undetected.
- ***SDD size***: the amount of added delay in nanoseconds (ns).
- ***Tested defect size***: The smallest SDD size at a fault site that is tested with the applied test clock and pattern.
- ***Effective SDD***: an SDD that is large enough to cause a failure in the circuit when operating at normal operating conditions, yet small enough to escape traditional TDF delay testing if tested with a short path under the same operating conditions.
- ***Smallest effective delay defect (SEDD)*** : The smallest size of an SDD at a fault site that can cause a failure in the circuit at normal operating conditions.
- ***Reliability defect***: An SDD that is smaller than the SEDD for the given fault, also known as a "timing-redundant" defect [20].
- ***True path***: a path in a circuit through which a transition can propagate to an endpoint under a possible sensitization condition (i.e. input vector).
- ***Test escape***: An SDD that escapes a TDF based delay test because it was not tested with the proper slack (defined by test path and operating conditions).
- ***Overtesting***: In this work, overtesting refers to testing an SDD of a size smaller than the SEDD of that fault.

The simplified example in Fig.1.2 shows the different types of delay defects that were defined above on a given fault site and for a given transition (rise or fall), where the paths that can carry the transition through this fault site are enumerated. As shown, a gross delay defect adds enough delay to be detected through any path. By contrast, an effective SDD detection depends on the path through which it was tested. On the other hand, a reliability

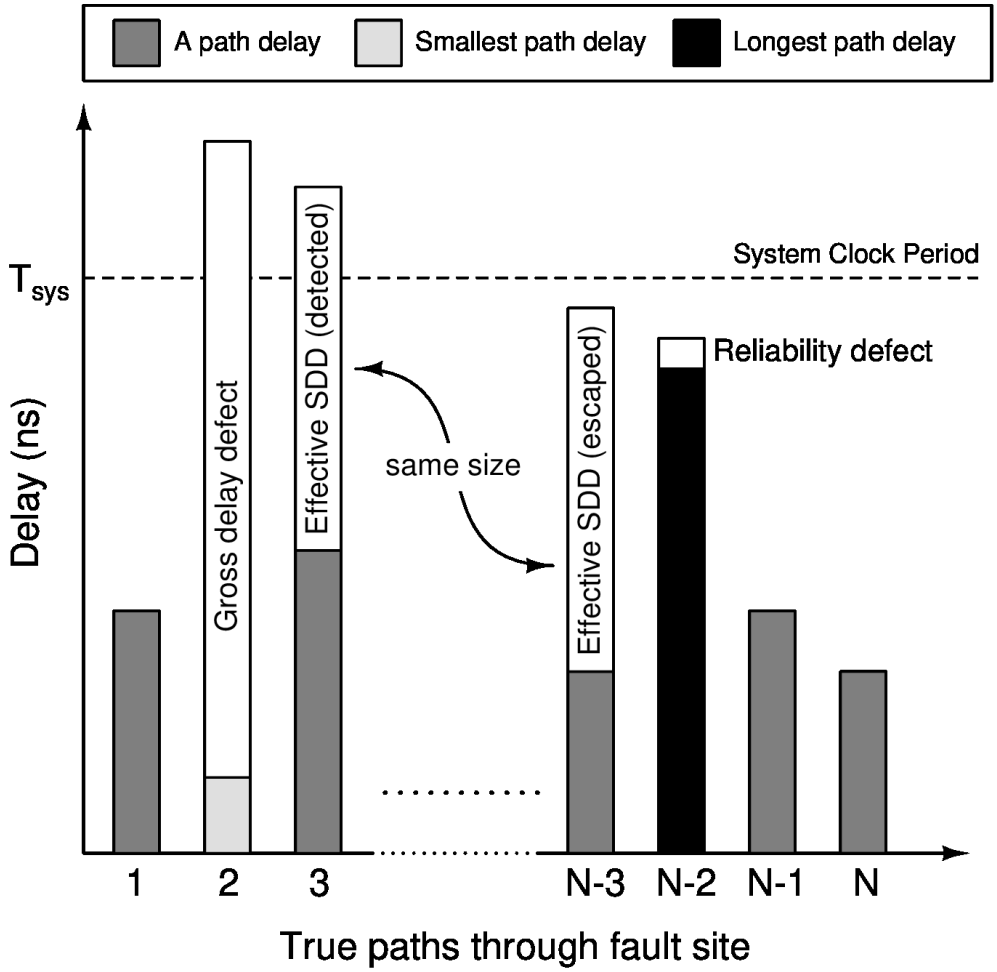

Figure 1.2 Example explaining the different types of delay defects with respect to path delays though a given fault.

defect cannot be detected with at-speed testing, no matter which path tests it. Remember here that we are following the TDF model in defining a delay fault, and by doing so, we are modeling there delay faults as a single fault located solely at a fault site (a node in a circuit).

The small size of SDDs creates many challenges in testing it properly. The main challenge in SDD testing is the ability to avoid false detection of a fault, and to accurately estimate delays during test pattern generation. Since the size of the defect is small, it is important to take into account factors that can induce delay in the circuit. For example, it is well know that for a digital circuit there is a relationship between the supply voltage and the delay of a digital gate [4,21]. If for any reason, the supply voltage fluctuates during testing, it can cause a small variation in delay that could be falsely identified as an SDD [22]. For example, a 10% variation on V_{DD} can create a 30% delay variation in 90nm technology and it gets worse in

more advanced technology nodes. For instance, in the 32nm technology the delay variation is 83% for the same 10% variation on V_{DD} [23,24]. In addition, cross-talk and process variation on a chip make the determination of the path delay and path lengths difficult, thus making the choice of the best path to test a fault through a very difficult one [25]. Delay variation on a chip (due to process, temperature or voltage variations) can also induce hazards (glitches) in a reconverging path. For SDDs, these hazards can mask the detection of a valid SDD or falsely flag a circuit as faulty [26]. It is quite challenging to develop a method to determine the validity of SDDs during testing, and obviously, the validity of a test result is definitely related to how similar the test environment is to the environment of normal system operation.

1.2.3 Scan Based Testing

As circuits complexity grows, it is important to gain access to as many nodes in a circuit as possible in order to test it properly. The most common way to apply delay test is using what is called *scan based testing* [27]. Scan based testing is a testing method where the inputs and outputs of the CUT are launched and captured through a scan chain. This method allows for the flexibility to inject test inputs and detect test results as needed in the CUT. In addition, because scan-based testing is integrated in a chip, it enables at-speed testing (testing at the same speed as normal operation). In contrast to trying to control the test speed using external testers that are usually built in an older (slower) technology.

In literature, there are two standard scan based delay testing methods: launch-on-shift (LOS), also known as *skewed-load* [28], and launch-on-capture (LOC), also known as *broadside* test [29]. The LOS is done by applying an initial vector (V1) and waiting for the signals in the CUT to settle before applying a second vector (V2). V2 is applied by shifting in one more bit and switching quickly from scan mode to functional mode, then issuing the capture clock. The disadvantage of this method is that the scan-enable (S_{EN}) signal needs to arrive at all the destination flops and settle before the capture clock edge occurs. This constraint becomes harder to meet as the test clock frequency increases (time between launch and capture decreases). On the other hand, the LOC does not have a timing constraint on the scan-enable signal. After applying V1, by scanning it in and waiting for the response to settle in the CUT, V2 will be ready at the inputs of the source flops waiting to be latched by the clock edge (launched). The test clock is pulsed twice, where the first pulse will launch V2 into the CUT, and the second pulse will capture the test result. The time between those two pulses define the speed of the test. The main limitation of LOC is the generation of V2. The classical approach is that V2 is the response from the CUT when V1 is applied to

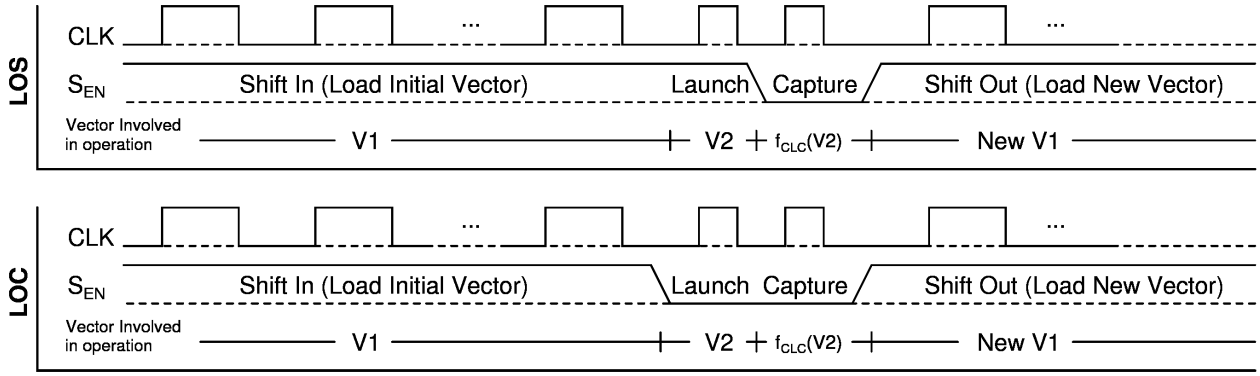

Figure 1.3 Timing diagram of the classical LOS and LOC.

its inputs. This makes the possible pattern combination limited and thus the test coverage would be lower than LOS. The timing diagram of the LOS and LOC is shown in Fig.1.3. Notice that due to the timing constraint on LOS, it is difficult to have the test run at-speed, whereas it is easy in the case of LOC.

1.3 The AnArm structure

In [1], Octasic presented a single-rail bundled-data handshake-free (self-timed) asynchronous processor architecture. Those technical words simply imply that there is a single control line (clock) and that the data has to arrive to the destination before the clock [3]. The Octasic architecture uses mainly flip-flops rather than latches (which is common in asynchronous designs) and it was successfully used in several generations of commercialized ICs. The architecture of [1] uses 16 execution units (EUs) to build an ad-hoc like processing pipeline. Every EU can be configured to process any instruction in the processor. The processing steps, access to resources and timing are managed by token rings. In this system, tokens are signals that circulate in token rings between all EUs. There is one token ring for each shared resource (e.g. data/instruction cache). Once a token is acquired by an EU, the EU holds the token and starts processing the part of the instruction requiring this resource (e.g. fetch, read/write register banks) [30, 31].

Fig. 1.4 presents a simplified schematic of an EU. When an EU acquires a token, a pulse is generated at node CLK_A. The pulse clocks the input flip-flops (source register) sending the data through the pre-configured combinational logic cloud. At the same time, the pulse travels through the delay line to the output flip-flops (destination register) and triggers the

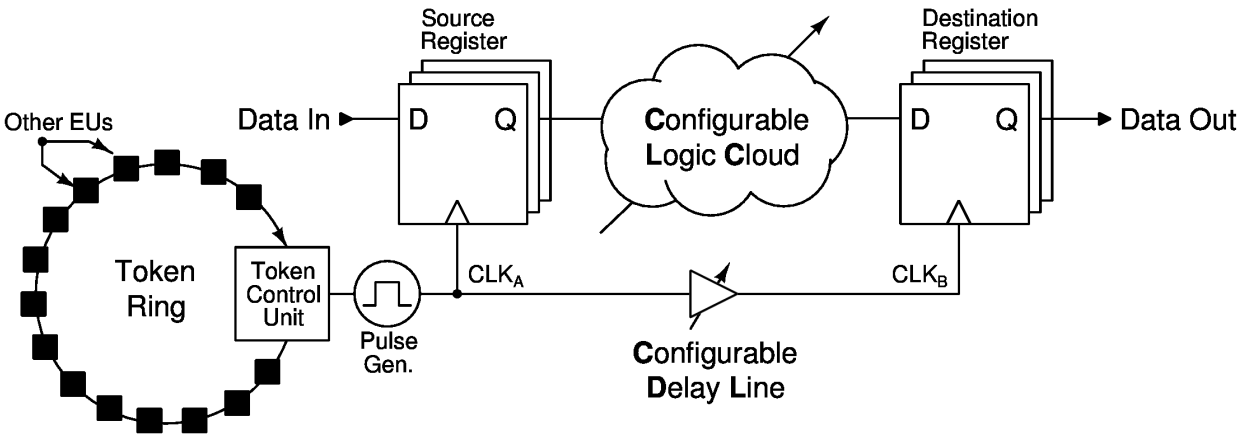

Figure 1.4 Simplified schematic of the execution unit (EU).

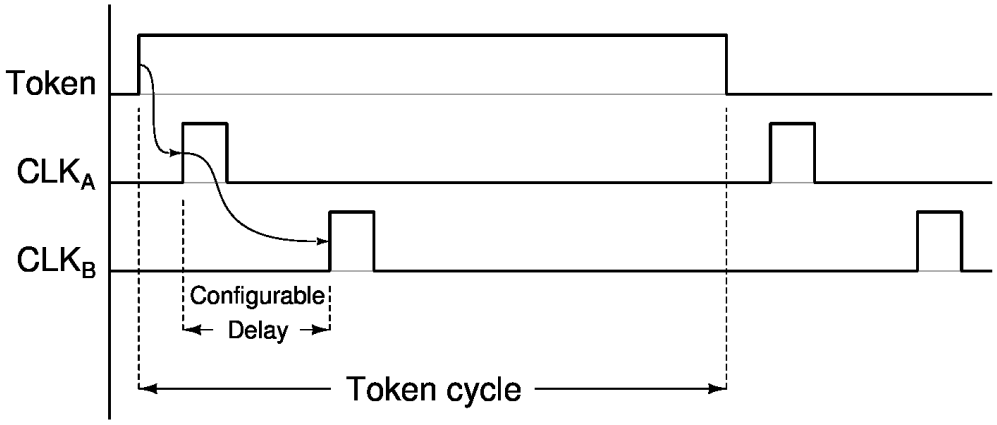

Figure 1.5 Timing diagram of the EU operation.

capture of the result of the processed data. Since the EU is a group of multiple different processing clouds, the delay of the delay line is configured to match the delay of the selected processing cloud. The selection of the processing cloud depends on the operation that is being processed (i.e. the type of the token acquired). The diagram in fig. 1.5 shows timing of the operation of the simplified EU.

Since the EU structure has a race between the clock pulse and the data, there are two timing constraints that need to be satisfied for it to function correctly [31, 32]: a setup time constraint and a hold time constraint. These constraints are similar to the synchronous circuits constraints. However, since this structure is self-timed and uses a configurable delay line (CDL), the setup and hold time constraints govern mainly the delay of the CDL with respect to the delays of the configurable logic cloud (CLC). The setup time constraints of the EU can be written as follows:

$$t_{CLC} \leq t_{CDL} - t_{setup} - t_{CK-Q} \tag{1.3}$$

Where t_{CK-Q} is the delay for the data to appear at the output of the flip-flop after the clock edge arrival, t_{CLC} is the combinational cloud delay, t_{CDL} is the delay of the CDL and t_{setup} is the setup time for the output flip-flops. As can be seen from this equation, to ensure proper operation, the delay of the CDL needs to be appropriately selected. For the hold time constraint of the EU, two conditions need to be satisfied:

$$T_{token} \geq t_{CDL} \tag{1.4}$$

$$t_{CLC} \geq t_{hold} + (t_{CDL} - T_{token}) - t_{CK-Q} \tag{1.5}$$

Where T_{token} is the time for the token to circle around the token ring and come back to the same EU, t_{hold} is the hold time for the output flip-flops and the rest of the terms are as previously described in equation (1.3). The first condition in the hold constraint governs the relationship between the token ring delay and the CDL delay. This relationship is usually satisfied by the fact that the token ring is much longer than the CDL. However, this puts an upper limit on the delay of the CDL. The second constraint is similar to the synchronous hold time constraint, however, since a single pulse launches and captures the data, a hold violation is very unlikely to occur in this architecture if the first condition of the hold constraint is satisfied. Hence, the delay testing strategy will focus solely on checking the setup time constraint by testing the delays in the logic cloud against the delay of the CDL.

1.4 Objectives

As we discussed in the previous section, Octasic presented a unique structure for a self-timed processor. The AnARM chip that is developed in this project is a testable version of the Octasic structure. It was designed and fabricated in the STMicroelectronics 28nm FD-SOI technology, and it is based on the ARM instruction set. As will be discussed in the next chapter, a scan-based at-speed test strategy for the AnARM have been developed as part of the team efforts on this project. However, the at-speed test method was not applied on the AnARM chips as the AnARM was still in the final design stages. In this work, it was our responsibility to test the developed at-speed test strategy an reuse it to develop a high-quality SDD test method.

To summarize, the following is the main objective of this work.

- Develop a high-quality SDD test method that targets the Octasic self-timed EU structure by taking advantage of its unique clocking style.

The following are secondary objectives that by achieving them we will achieve the main objective:

- Propose an SDD test model and test quality metric that can take into consideration the unique clocking style of the EU.
- Develop an algorithm that adapts the CDL delay during pattern application to maximize the proposed SDD test quality metric.
- Validate previously developed at-speed test method and use the existing test structures to apply the developed high-quality SDD test.

The following are side objectives that are in-line with this work but have only been partially achieved due to the time limitation of this project:

- Study, and if possible include, the effect of PVT variations during the development of the SDD quality metric.
- Study the pros and cons of the Octasic EU clocking style, compared to classical synchronous clock trees, in terms of SDD testing.

Each of these objectives led to the proposed solutions and methods that are claimed as contributions. **Considering the unconventional characteristics of the Octasic design style and the required literature background to understand the contributions of**

this work, we elected to fully state the contributions of this work, in section 2.7, after fully reviewing prior works done in SDD testing and delay testing on the AnARM.

Finally, note that although the objectives of this work target the Octasic structure, the presented work and contributions can be expanded to conventional synchronous systems. For this reason, many of the coming sections do not specifically target the Octasic structure.

1.5 Work Organization

This dissertation is organized as follows. In chapter 2, the contributions of this work will be detailed after a review of literature. A novel SDD test model, and a SDD test quality metric that is derived from it, are presented in chapter 3 along with simulation results and discussions. Chapter 4 presents a method of building an optimized SDD test using the developed SDD test quality metric. Simulation results proving the validity of the presented technique are presented in the same chapter. The applied test results on silicon are presented and discussed in chapter 5. Finally, conclusions and recommendations for future work are listed in chapter 6.

Note that some parts of this work generate great amount of data and figures. In order not to distract the reader with too many figures, supplementary figures have been added in the appendices.

CHAPTER 2 PRIOR WORKS ON SMALL-DELAY DEFECT TESTING AND ANARM AT-SPEED TESTING

As mentioned in the objectives (in section 1.4), the goal of the work presented in this dissertation is to develop a high-quality SDD delay defect test for the AnARM processor. In order to assess the contributions of this work, this chapter will present the most notable works done in the domain of SDD testing, and prior works done on developing a delay test for the AnARM. Then, at the end of this chapter, we conclude with a list of the contributions of this work.

Before proceeding with the review of prior works, it is important to note that in the literature, the term SDD is loosely defined as any delay defect that is small enough to escape delay testing. In order to be clear, we will indicate, when possible, the specific type of SDDs that the reviewed work is talking about with reference to the definitions previously listed in section 1.2.2.

2.1 Overview on SDD Testing

As mentioned in chapter 1, standard delay testing is based on one of two delay fault models: the path delay fault (PDF) model or the transition delay fault (TDF) model [8, 12, 13, 27]. A PDF based delay test is an extensive search of all possible paths in a circuit for delay defects. Effective SDDs cannot escape detection if all paths in a CUT are tested with an at-speed speed clock under normal operating conditions. However, as the circuit complexity grows, the time to apply such a test can grow exponentially, and the conditions to correctly sensitize paths become more complicated [16, 18]. On the other hand, a TDF based test applies test vectors to excite a transition (rising and falling) through each node in a circuit. It is less time consuming to apply a TDF based test than a PDF one. However, since not all paths are tested, it is possible to miss an SDD that can cause a chip to fail in the field.

To test for SDDs, most researchers have focused on building tests that enhance TDF based tests. The main idea in the majority of the literature is to reduce the slack window in which an SDD can escape during testing. This is either done by enhancing the ATPG algorithm to sensitize longer paths (in terms of delay) [17, 26, 33–44], or by applying a test clock that runs faster than the normal system clock [45–52]. In this case the test is referred to as faster-than-at-speed testing (FAST).

2.2 Pattern Generation for Small-Delay Defect Testing

An ATPG is a software that is used to generate the stimulus (*vectors*) that test the CUT and define the expected results [4]. In the classical stuck-at fault model, only one vector is needed to test if a node is stuck-at a logical 0 or 1. In delay testing, a transition is needed to check for delay faults and so a *pattern*, which consists of two vectors, is needed. The group of all patterns generated to test a circuit is called a *pattern set*. Under the standard TDF model, these patterns are generated without considering path delays. Thus, as mentioned in the previous subsection, TDF based delay tests cannot detect effectively SDDs. Some researchers in the field of delay testing tried to enhance the ATPG techniques to address this problem by generating better delay test patterns.

Looking at the research that was done on SDD testing from the side of pattern generation and selection, we see that all the ideas are focused on generation and/or selecting patterns that reduce the slack the path through which a fault is tested [17, 33, 35–37, 39, 42, 53]. In this field of research there are two schools of thought: one is to construct a timing-aware ATPG (TAA), and the other is to use other non-timing driven techniques that reduces the probability of a test escapes.

Starting with the TAA, this method revolves around finding a path through the fault site which has the least slack [17]. This requires that the tool computes the delay of gates and metal lines and searches for the path with the least slack. The computation effort of this approach is clearly more than the standard timeless ATPG algorithm but it results in a better selection of the test vectors. To avoid the complexity added by a TAA, some methods were developed to detect SDD under the TDF model without calculating the longest path through a fault. The n-Detect technique is one of the techniques that can be used for enhancing the SDD test under the TDF using timeless ATPG [53]. The idea of the n-Detect method is simple; if one finds multiple patterns that activate the same fault site multiple times through different paths, then the probability that one of the selected paths is the path with the least slack is the higher. The trade-off here is obviously between the number of the test vectors and the probability of detecting SDDs. Another approach is to assume that the longest structural path is the path with the least slack. For example, the "As Late As Possible Transition Fault" (ALAPTF) technique selects test vectors that sensitize the longest structural path from the input to the fault site [37]. Likewise, another technique, called K-longest path, is an approach that activates each fault site with the longest structural paths from the fault site to the destination flops [40]. Of course the disadvantage of the longest

structural path techniques is that there is no guarantee that the longest structural path is the path with the minimum slack.

2.3 Dealing with PVT variations

In the previous chapter we mentioned that one of the main challenges with SDD testing is the presence of delay variations during testing. PVT variations and cross-talk can make the correct identification of SDDs difficult. Several works in the literature have proposed methods for dealing with delay variations. For example, in [55], the authors proposed a test flow (for testing effective SDDs) that feeds delay information from the chip back to the tester in order to readjust the test accordingly. This method requires that a delay measurement structure in the chip is able to communicate with the tester, and that the tester is able to adjust the test quickly. Else, this process becomes a very lengthy process of adapting the test to each individual chip. Another work looked into detecting SDDs in the presence of process variations by comparing the delay of inter-correlated paths [56]. If the delay changes in one path but not in the correlated path, then an SDD is flagged. This method only targets a limited number of critical paths in a circuit.

One of the difficult aspects of testing SDDs in the presence of process variations is the ability to distinguish an SDD from a delay induced by process variations. The work in [41,54] is frequently cited for defining a *detectable delay defect* as a defect that has at least 50% probability of detection in the presence of process variations. The authors of [54] model a path delay as Gaussian distribution with a mean of μ and standard deviation of σ. They also define that a delay defect is detectable if the added delay is beyond the 3σ of the fault-free path distribution. It is assumed here that a delay defect will only shift the mean of the path delay distribution, and that the probability of detecting a defect is based on the area under the faulty-path delay distribution that crosses the fault-free 3σ threshold. This is depicted in Fig. 2.1. In summary, the probability of detection is 50% when the size of an SDD is 3σ. This defines the size of the smallest detectable delay defect in the presence of process variations. This definition is the base for other works in the literature (such as [45, 56]) that set the size of the SDD that needs to be tested to 3σ of the fault-free path delay distribution. Note that however, depending on the system clock period and path delay distribution of the overall circuit, this size of delay defects may fall under the reliability defect size definition.

To deal with variations induced by cross-talk and process variations, [25] studied the effects of cross-talk between paths on delay variations using H-spice. The results show that a

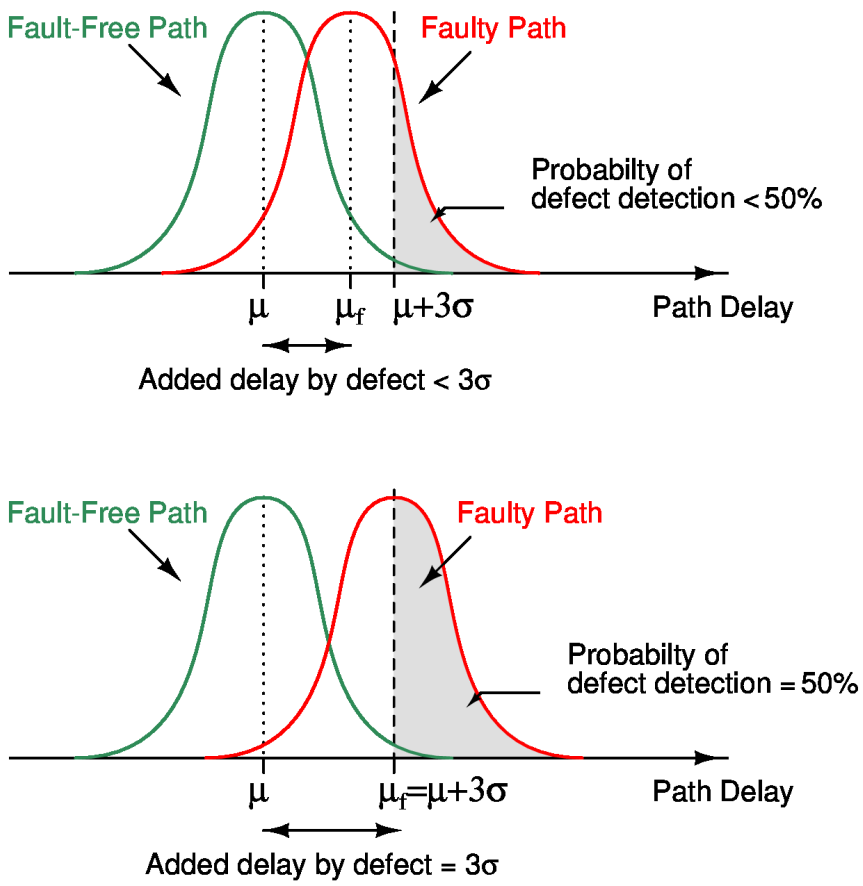

Figure 2.1 Example of probability of delay defect detection as defined by [54].

transition can speed up, if two neighboring paths have the same direction of transition, or slow down, if the direction is the opposite. Using Monte-Carlo analysis, process variations are considered as well. Paths are assumed to have a Gaussian delay distribution with process variations. The delay induced by cross-talk is modeled to affect only the mean value of the computed path delay distribution. In [25], patterns are generated through various algorithms and filtered based on a threshold that is defined on a pattern weight factor. This pattern weight factor is defined as:

$$W_{pat_i} = \sum_{i=1}^{M_i} W_{path_i} \tag{2.1}$$

where M_i is the total number of long paths sensitized with pattern i and W_{path_i} is the weight of each path. This weight is calculated based on threshold (LP_{th}) that is related to the system clock period T_{sys}, and the delay distribution of a path. The example in Fig. 2.2 shows how W_{pat_i} is calculated. This flow is a complex one. Applying this method involves Monte-Carlo

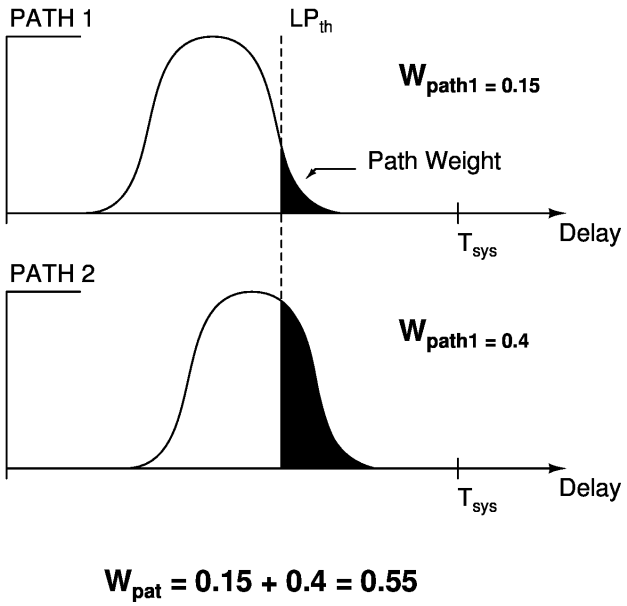

Figure 2.2 Example of pattern and path weight calculation based on [25].

simulations and coupling capacitance extraction from the circuit layout.

For the effect of power variations on SDD testing, [22, 49] developed a method of filtering TDF patterns based on the circuit activity created by the pattern during testing. Excessive activity causes the power supply node to droop and the ground to bounce. This creates delay changes in the paths of the CUT. To avoid that, the authors of [22, 49] proposed a pattern filtering metric called the *switching cycle average power* (SCAP). This metric is based on the dynamic power and switching activity of the CUT. Patterns that create heavy activity and high demand on power are rejected.

2.4 Faster-Than-At-Speed Testing (FAST)

Choosing the test clock frequency is a very important part in designing a delay test. Since the test clock frequency is related to the available slack on a path, and thus to the size of the tested delay defect, it is one of the factors that determines the quality of a delay test. Running the test at-speed (the speed under which the system normally operates) or faster-than-at-speed is highly desirable in order to increase the quality of a delay test. By reducing the slack of the tested fault using a higher frequency test clock, FAST can enable TDF based testing to catch SDDs [48, 49].

There are some challenges that needs to be addressed when using FAST. The first challenge is in generating the high speed test clock. Normally, automatic test equipments (ATEs) cannot be used to run higher frequencies due to the limitation on either the signal integrity of off-chip signals, or simply because the ATE is built with an older technology than the chip under test, and thus, runs at a slower speed than newer chips. Due to these limitations, the test clock needs to be generated from within the chip. This normally requires phase-locked loop (PLL) structures that can generate different test clock speeds (as needed). The other challenge is related to the consequences for running the CUT with a high clock speed. One of the these consequences is the elevated power drop effects due to the fast switching clocks. These power drops can result in false test rejects, as well as added power supply noise due to faster switching [49]. Running at a higher frequency can also mean the possibility of damaging the chip due to effects such as overheating, not to mention that the circuit should be able to function properly under those elevated speeds. Even with these challenges, it can be advantageous to test SDDs with FAST. It was demonstrated in [41] that longer paths are more susceptible to process variations. Thus, testing delay faults with shorter paths and faster test clocks reduces the effects of process variations on the test.

Since applying FAST entails that some paths exhibit negative slacks, special care has to be taken to avoid test on these paths. In addition to the obvious requirement of masking endpoints with negative slacks, works in FAST has handled this issue in various ways. For example, [49] grouped patterns based on the maximum delay of the sensitized paths, then they assigned the appropriate test clock frequency to each group (Fig.2.3). Since changing the frequency of the test clock requires re-adapting the frequency of the PLL, an increases in the overall test time is to be expected. However, by grouping the patterns before testing, all the patterns that result in similar maximum delay are applied under the same clock speed. This method helps minimizing the jumps between different clock frequencies and thus minimizes the overall test time.

If the test clock speed is faster than some of the data paths, the circuit could still have some signals switching while the test results are being captured. In this region of signal activity, changes in the data through the circuit can create glitches that might cause the CUT to be falsely labeled as defect free or a reject. One way to guarantee that no glitches would occur on a path is to check for the robustness of the tested path. This can be computation intensive. In [26] a FAST method was specially developed to be applied deep into the region of activity to test very small defects on short paths. This is done without computing path robustness, but rather through a simulation based approach that takes into account process

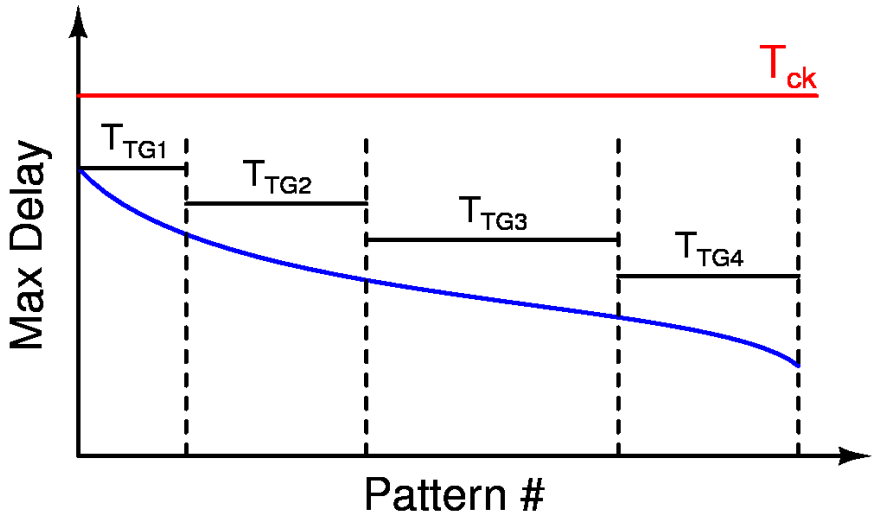

Figure 2.3 Grouping patterns based on the maximum delay achieved with each pattern and selecting the appropriate test clock period [49].

variations. A regular TDF pattern set is tested by checking the result of transitioning with a defined uncertainty region (labeled X). If a hazard is appears due to the injection of X, then the pattern is rejected. The result is a hazard-free pattern set that is used for detecting SDDs.

2.4.1 Programmable Clock Generator for FAST

As we mentioned earlier, one of the challenges in applying FAST is in generating the needed clock frequencies. In contrast to the in inflexible solutions that uses PLLs, a programmable clock generator was presented in [47] that enables synchronous circuits to apply FAST with classical scan-based timing schemes using buffer chains controlled by registers. This clock generator can be also used as a delay measurement circuit for path delays in the CUT. One of the interesting attributes of this solution is the ability to program the test speed as part of the applied pattern. This can reduce the test time and removes the need to group patterns by the maximum activated path delay.

We focus on this solution particularly because of its similarity to the Octasic clocking structure. The clock generation circuit in [47] uses a delay line to generate faster-than-at-speed clocks. The clock delay in the Octasic structure can be used in a similar way, and so is the ability to program the test speed as part of the test. The difference however as we will emphasize later, is that in the Octasic structure, the CDL is distributed whereas in [47] the

programmable clock generator is centralized. That means that, in the Octasic structure, we can potentially test different circuits with different test clock speeds at the same time.

2.4.2 FAST optimization

When building a FAST delay test, there are three elements with which the test can be adjusted: the pattern set, the clock speed and the output (endpoints) masking. This fact brings up the opportunity for a method of optimizing FAST. Since it is impossible to generate unlimited numbers of test frequencies, there needs to be a way of selecting an optimum set of frequencies that maximizes the quality of a test. A method of selecting faster-than-at-speed clock frequencies for a fixed SDD fault size was presented in [50,57,58]. By assuming a known fixed size of an SDD, the authors of [58] define detection ranges for each fault by simulating the CUT with the given pattern set. They then apply the hypergraph algorithm that is presented in [59] to generate the minimum set of test clock frequencies needed to detect all targeted SDDs.

Alternatively, [45] presented a method of optimizing FAST by searching for short paths that can be sensitized under the PDF single path sensitization criteria while covering all TDF faults in the CUT. The single path sensitization criterion requires that static non-controlling values be applied to all side-input of a path when a transition is propagated to the output. It is tighter than the robust sensitization criterion, and only one transition can be propagated to the output at a time. This is why the targeted paths have to be short paths. This kind of test requires that the faster-than-at-speed clock be fast enough to test these short paths. The authors here use the 3σ defect size as the size of the minimum SDD to be tested, and they redefine overtesting according to this size. Shorter paths, as mentioned earlier, are less prone to process variations. However, the single path sensitization criterion could lead to higher pattern count, and the selected 3σ limit means that many of the tested SDDs are in the reliability defects size range.

Lastly, [52] proposed a FAST optimization technique that reduces the tested slack for each fault while preserving the TDF coverage and limiting the growth in the pattern size. Three optimization options were presented: test slack minimization, test path delay maximization and pattern count minimization. All these optimization methods try to reduce the test slack as much as possible to catch the smallest SDD on the tested paths. As long as the path slack is not negative, the test path and test frequency pairs are considered in the test. Masking of negative slack endpoints are applied in with these optimization strategies as needed.

As a final note, we would like to point out that, in general, FAST in literature try to assess the reliability of the chip by searching for all sizes of SDDs (even if they are too small) since they have the potential to grow as the chip ages [60]. In this work, however, we are more interested in effective SDD that escape TDF based delay testing.

2.5 Small-Delay Defect Test Quality Metrics

Metrics are important as they define the quality of a delay test. SDD test quality metrics are used to assess the effectiveness of a delay test in catching SDDs. The quality of a classical TDF based delay test is measured by the transition delay fault coverage (TDFC). TDFC is merely the percentage of tested nodes to total number of nodes in the CUT. Since the TDFC does not depend on the size of the tested delay defect, it is not adequate for representing the quality of SDD tests. That is why many researchers use other test quality metrics that they normally report along with the TDFC, when presenting new work in SDD testing.

The metrics presented in [17, 20, 61–64] are some of the generic SDD quality metrics that are found in the literature, and some them are used in well-known test software. In general, these SDD test quality metrics can be divided into two categories: statistical and non-statistical. Statistical metrics require the delay defect distribution for the target fabrication technology. This distribution shows the probability that a defect of a certain size exist in a given technology. It is used to calculate importance of catching a delay defect of a certain size. Although the idea of using a delay defect distribution is compelling, it is often impractical. Such a distribution is not provided by manufacturers as part of the PDKs, and all the current statistical metrics have used the same distribution extracted from empirical data that are a result of an extensive test of more than 70,000 chips fabricated in the 180 nm technology [65]. Thus, without an accurate delay defect distribution specific to the technology and the foundry used, no added value comes from using such distribution. Moreover, no information on the variations related to power or temperature are extracted from the experiment in [65]. So, the accuracy of this distribution is also limited to the typical voltage and temperature conditions. On the other hand, non-statistical metrics are simpler to calculate and do not require a delay defect distribution. Obviously, non-statistical metrics do not consider the probability of occurrence of certain sizes for delay defects in a technology when evaluating the quality of the SDD test, nor do they have any information on PVT variations. As a consequence, for a single path in a circuit, as long as the ratio of the tested delay (tested slack) to the longest delay (smallest slack) is constant, the value of those metrics will not change.

We will review in this section the most frequently cited metrics in the literature and discuss them briefly. To minimize redundancy with later parts of the dissertation, we will define here some of the commonly used terms. Firstly, from here on, the term *metric* will be used to mean a *SDD test quality metric*. Note that all the metrics that are discussed in this dissertation are for TDF based tests. As defined earlier, under the TDF model, a *true path* is a functional path that a transition can travel through starting from a circuit input (primary input or scan flip-flop out), passing through the targeted fault site and arriving at a circuit output (endpoint, namely, a scan flip-flop input). For a given fault (ϕ_i), we will denote the delay of the longest true path passing through a fault as PD_{LT_i}, whereas PD_{LA_i} will denote the delay of the longest activated (testable) by a test pattern and passes through a fault. Also, T_{sys} will denote the period of the system clock and T_{test} will denote the period of the test clock.

2.5.1 Delay Test Coverage (DTC)

The *Delay Test Coverage (DTC)* is a simple non-statistical metric that, consequently, does not use a delay defect distribution function [17]. The DTC is basically calculated as the average ratio between PD_{LA} and PD_{LT} for each fault in the circuit:

$$DTC = \frac{1}{N}\sum_{i=1}^{N}\frac{PD_{LA_i}}{PD_{LT_i}} \times 100\% \tag{2.2}$$

where N is the total number of faults in the CUT. Equation (2.2) shows that the DTC does not consider the test clock frequency, the slack of the path nor the delay defect distribution. It implicitly assumes that all delay defect are equally likely to happen regardless of their size. Although this makes the DTC not representative of the probability of a delay defect occurrence, it is fast and easy to calculate. Moreover, since the DTC does not use any slack measurement in the calculation, it is only accurate when the test clock period and the system clock period are equal (i.e. at-speed testing), and therefore, it cannot be used for FAST.

2.5.2 Statistical Delay Quality Level (SDQL)

The *Statistical Delay Quality Level (SDQL)* [20] is calculated based on the probability of SDDs escaping the delay test (test-escapes). This probability is calculated under the *Statistical Delay Quality Model (SDQM)* to be the range of SDD values that are not covered by a test pattern set when applied with a test clock period. To quantify the probability of having a test-escape, a delay defect distribution is extracted from the probability of having a delay defect in a certain technology, which is then fitted using an equation of the form of $F(s) = a\,e^{-\lambda s} + b$ (Fig.2.4). The SDQM defines two slacks: the test slack margin S_{mgn} and

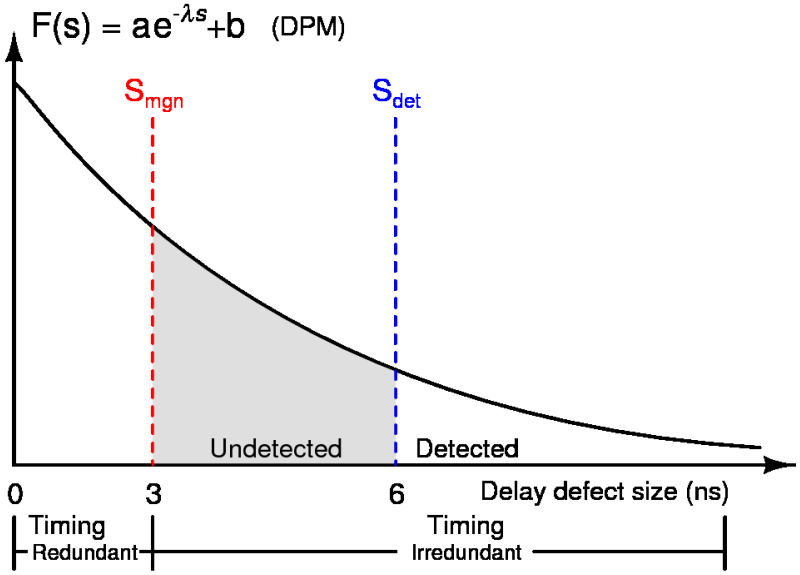

Figure 2.4 Example of delay defect distribution divided into regions based on the value of S_{mgn} and S_{det} [20].

detection slack S_{det}. These slacks are calculated as follows:

$$S_{mgn_i} = T_{sys} - PD_{LT_i} \tag{2.3}$$

$$S_{det_i} = T_{test} - PD_{LA_i} \tag{2.4}$$

As discussed in [20], S_{mgn} splits the delay defect distribution into two regions. Any defect smaller than S_{mgn} is called a timing redundant delay defect, which is a reliability defect. Whereas delay defects larger than S_{mgn} are considered as effective SDDs (timing irredundant). S_{det} splits the timing irredundant region into two subregions. Delay defects that are larger than S_{det} are detected by the test, whereas those smaller than S_{det}, but larger than S_{mgn}, are considered as undetected (test-escapes). Under those definitions, the SDQL for each fault is calculated as the area under the curve between the two slacks (as indicated by the shaded area in Fig.2.4). For the complete set of faults in a circuit, the SDQL it is calculated as:

$$SDQL = \sum_{i=1}^{N} \int_{S_{mgn_i}}^{S_{det_i}} F(s)\, ds \qquad \text{(DPM)} \tag{2.5}$$

where N is the total number of faults in the CUT, $F(s)$ is the *delay defect distribution function of a delay defect of size s* and the unit for the SDQL is defect per million (DPM).

The SDQL targets undetected timing irredundant delay defects. Thus, it does not include timing redundant delay defects into the calculation and the lower the SDQL the better. This metric has two shortcomings. First, because the SDQL is not a normalized number it gives an unfair advantage to circuits with a lower number of faults over those with a higher number of faults [62]. This also means that it is difficult to know what is a good level of SDQL for a circuit without a reference. Second, in FAST, it is possible for S_{det} to become less than S_{mgn}. In that case it is not clear how to calculate this metric, however, in our computations, we assign a perfect score (of 0) to this case.

2.5.3 Small Delay Defect Coverage (SDDC)

Another statistical metric is the *Small-Delay Defect Coverage (SDDC)* [62]. The SDDC measures the ratio between the probabilities that a defect is detected by the test and the probability that a defect is timing irredundant (can cause a failure during normal operation). In reference to Fig.2.4, the SDDC can be calculated as follows:

$$SDDC = \frac{1}{N}\sum_{i=1}^{N}\frac{\int_{S_{det\_i}}^{\infty} F(s)\,ds}{\int_{S_{mgn\_i}}^{\infty} F(s)\,ds} \times 100\% \tag{2.6}$$

where N is the total number of faults and $F(s)$ is the *delay defect distribution function of a delay defect of size s*. In the case of faster-than-at-speed testing, if the SDDC term for any fault is > 1, SDDC is split into three parts: 1) The normal SDDC metric; 2) SDDC_{DPM}, which is the same as SDDC except that the score of each fault is limited to a max of 1; 3) to represent the percentage of reliability testing (testing of timing redundant faults), SDDC_{EFR} is defined as the difference between SDDC and SDDC_{DPM}. The disadvantage of such definition is that faults that are over-tested (result > 1) are given a perfect score (of 1) in the SDDC_{DPM}. This can be a misleading result if one is not interested in reliability defect, because overtested faults can lead to false rejects (i.e. chips falsely identified as faulty). Moreover, as the case with previous metrics, the SDDC does not take into consideration any delay variation due to PVT variations which can cause inaccuracies in fault identification and slack measurements.

2.5.4 Quadratic Small Delay Defect Coverage (SDDC^Q)

The *Quadratic Small-Delay Defect Coverage ($SDDC^Q$)* is a non-statistical metric that was presented in [62] as an alternative to the statistical SDDC when a delay defect distribution

is not available. The SDDC^Q is calculated with the following equation:

$$SDDC^Q = \frac{1}{N}\sum_{i=1}^{N}\frac{(PD_{LA_i} + T_{sys} - T_{test})^2}{(PD_{LT_i})^2} \times 100 \tag{2.7}$$

where N is the total number of faults. Notice that the SDDC^Q takes into account the period of the test clock and system clock that is normally used. It was shown in [62] that SDDC^Q is proportional to the delay defect size. This gives it an advantage over the DTC. However, it is difficult to interpret the significance of the SDDC^Q equation. The results in [66] have shown that SDDC^Q can sometimes report invalid values ($\gg 100\%$) when T_{test} is not equal to T_{sys}. This makes it unreliable for use in FAST.

2.5.5 Statistical Delay Fault Coverage (SDFC)

The *Statistical Delay Fault Coverage (SDFC)* [64] is a metric that takes into account the delay distribution of the longest delay path passing through a fault site, the test clock frequency and the probability of having a delay defect in a path. To calculate the SDFC, we need to calculate what is referred to as the *system sensitivity (S)* to a fault:

$$S = P[W > T_{CLK}] = \int_{t=T_{CLK}}^{\infty} f_W(t)dt \tag{2.8}$$

$$f_W(t) = f_X(t) * f_Y(t) \tag{2.9}$$

where W, X and Y are random variables with probability density functions of $f_W(t)$, $f_X(t)$ and $f_Y(t)$ respectively. Y represents the probability of delay on a certain path in a good circuit, X is the probability of have a defect in a path and W is probability of a delay in a faulty circuit. Notice that the probability density function of W is the *convolution* of the X and Y probability density functions. A simple graphical example of calculating the system sensitivity of a path is presented in Fig.2.5.

Then the *individual test effectiveness* is defined as (E):

$$E = \frac{S'}{S} \tag{2.10}$$

where S' is the system sensitivity for the *activated path through the fault site* and S is the system sensitivity of the *longest functional path through* the fault site. Now the SDFC can be defined as the ratio of the sum of system sensitivities of the longest path tested by the test vector over the sum of the sensitivities of the longest possible functional paths passing

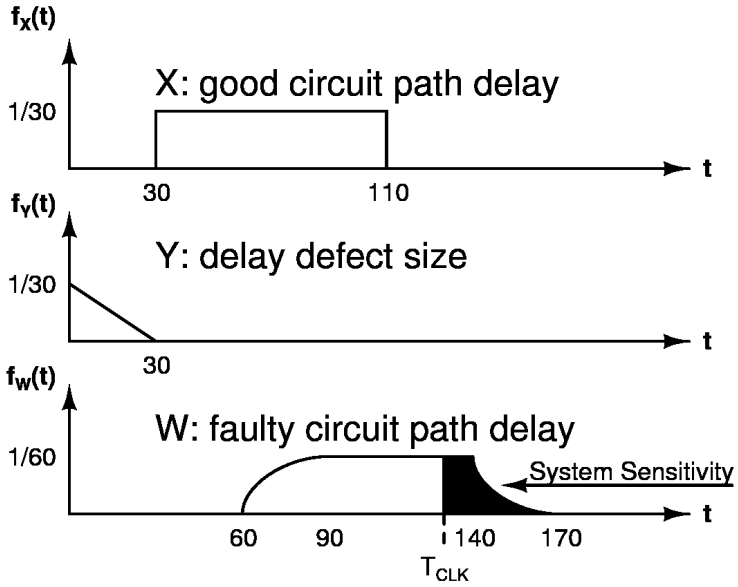

Figure 2.5 A small example on the calculation of the system sensitivity.

through a fault site:

$$SDFC = \frac{\sum_{i=1}^{N} S'_f}{\sum_{i=1}^{N} S_f} \times 100\% \tag{2.11}$$

where N is the total number of faults in the CUT. Notice the definition of the system sensitivity cannot be applied to test configurations where the test clock is not running the same frequency as the system clock.

2.5.6 Metrics Summary

In summary, all of the presented metrics in this section are used for SDD testing. The DTC is a simple metric that is easy to calculate but that does not take into account the frequency of the test clock. This makes it an inaccurate measure when the test clock is not running at the same frequency as the system clock. It also does not take into consideration the delay defect distribution. This means that it considers defects from all size to have the same probability of occurrence, which is not the case in CMOS technologies. The statistical metrics SDFC, SDQL, SDDC do include the delay defect distribution information into the metric but this distribution is not normally supplied with a design kit and it needs to be estimated by a third party for a given process. Therefore the delay defect distribution is only an estimation and it is not trivial to get. Notice that none of the proposed metrics discusses the case of FAST except the SDDC.

Lastly, remember that these metrics are used for SDD testing of digital circuits. The standard flow of digital circuit requires simulating the circuit performance under multiple PVT points to ensure the proper operation of the circuit under various operating conditions. This raises the question of where in the PVT space would one plan the SDD test? The discussed metrics assume a single PVT point when extracting the path delay information. Can these metrics be used with multiple PVT point flow? We will explore these questions in chapter 3.

2.6 At-Speed Testing of the AnARM

At-speed delay testing is important to guarantee the correct behavior of circuits when operating at the rated speed. There are many different ways of data propagation in asynchronous systems [3], and whether delay testing is required or not depends on the type of asynchronous circuit being tested [67]. For example, most high-speed asynchronous designs require delay testing, since they have to satisfy some timing conditions in order to operate correctly [68].

It is difficult to standardize the design and test process of asynchronous circuits due to their diverse design styles. This makes it challenging to use conventional design tools that are normally used for synchronous circuits [3]. Part of the difficulty stems from the fact that several asynchronous design styles are based on custom C-elements. In addition to the hassle of using custom standard cells for C-elements, applying conventional testing methods on circuit that adopt such asynchronous design styles is not trivial. Several papers in the literature have proposed methods for testing circuits designed according to such asynchronous design styles [67–70] by modifying C-elements and latches to apply the required test.

To apply at-speed testing on asynchronous circuits, it is essential to use a test clock speed equal to the one used during operation. This demands a careful study and effective reuse of the timing mechanisms of the asynchronous structure during at-speed testing. Generally, the global asynchronous system speed can be inferred in several ways. However, the exact speed of operation of internal asynchronous clocks after fabrication cannot be predicted due to the effects of PVT variations on circuit delays. Thus, the use of external test clocks with predefined frequencies by an ATE is not adequate in this case.

As previously mentioned in section 1.1, the AnARM is a testable self-timed processor that uses the Octasic design structure. The test structures that are inserted in the AnARM

processor allow us to apply stuck-at testing on most of the design and at-speed delay testing on a specific part of the processor. The work in this dissertation reuses the at-speed structures and the modified LOC strategy that is in place to apply SDD testing by adding additional delay stages to the CDL. In this section the method of applying at-speed testing will be reviewed, and later in chapter 5, the silicon test result that for both the at-speed test and the proposed SDD test will be presented.

The AnARM at-speed test strategy applies at-speed test clocks by taking advantage of the built-in CDLs while using conventional scan-based testing. The idea is to first preconfigure the logic cloud and the delay line, then scan-in the test vectors using a synchronous test clock, then use the same clock to trigger at-speed launch and capture of the test vector using the CDL, and lastly scan-out the result. Note that the test structures that are put in place are more complicated that what is presented here. For simplicity and clarity, these test structures have been reduced to show only the effective part used in at-speed testing.

2.6.1 Test Structures

Fig. 2.6 shows a generalized view of the AnARM self-timed structure. Depending on the number of stages, this generic structure can be used to represent multiple parts inside the AnARM. In this structure, the data travels from one stage to another, and a clock pulse, traveling through the delay lines, controls the launch and capture of the data from one stage to another. Notice that only one pulse would travel down the multi-stage path at a time, and the delay of the CDL is preconfigured appropriately to test the related logic cloud. This should not be confused with how the processing pipeline is constructed in the Octasic architecture with multiple EUs.

In order to apply at-speed testing for this architecture, two types of test structures need to be added: scan chains and test clock insertion logic. One of the advantages of this self-timed architecture is that it does not depend on C-elements nor latches. Hence, there is no need to design special standard cells for C-elements, nor there is a need to develop new methods for scanning the data or use extra clocks (as in the L1L2* methods [69, 70]). This enables the design to use regular scan flip-flops, exactly like synchronous architectures, where the scan flip-flops have scan-enable (S_{EN}) and scan-in (S_I) signals and where the input/output data scanning operation is driven by a synchronous test clock.

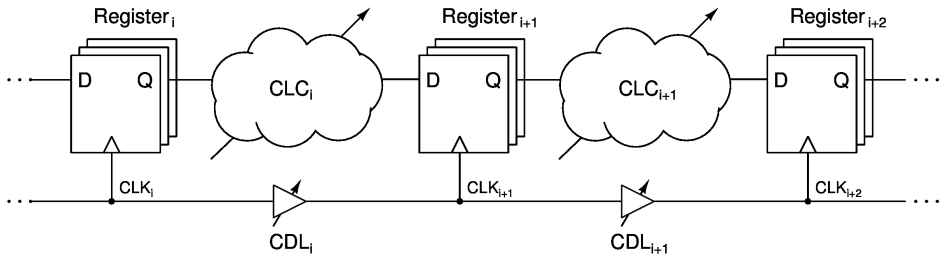

Figure 2.6 Generalized multi-stage self-timed structure using the Octasic design style.

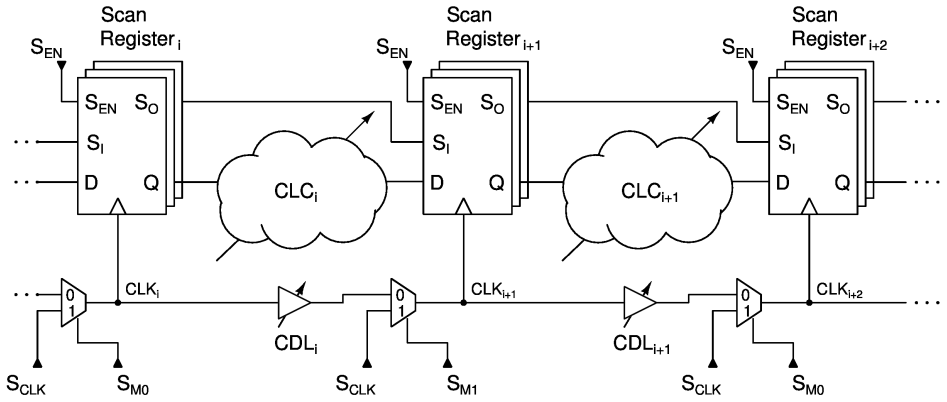

Figure 2.7 Test structures inserted into the targeted multi-stage self-timed structure.

A test clock insertion and management circuit is added to feed the synchronous test clock into the system and to control the process of switching between different clocking schemes. This is achieved in the AnARM with a simple multiplexer. Fig. 2.7 shows the simplified multi-stage self-timed structure after the test structures have been inserted. The two main changes are as follows:

1. Normal registers are replaced by chained scan registers.
2. A multiplexer is inserted just after each CDL to allow synchronous clocking (using S_{CLK}) with interleaved scan mode signals (S_{M0} and S_{M1}).

2.6.2 Test Modes

The AnARM testable structure (seen in Fig. 2.7) allows for one of three different modes. Firstly, the regular operation mode (mission mode) is configured by setting all the test control signals (S_{EN}, S_{M0} and S_{M1}) to 0. Fig. 2.8 highlights the activated paths during mission mode with a thick line. The added multiplexers do not add much delay to the system. Moreover, both the data path and the clock path have a 2-input multiplexer added to them. The one on the clock path is obvious, while the one in the data path is built into the scan flip-

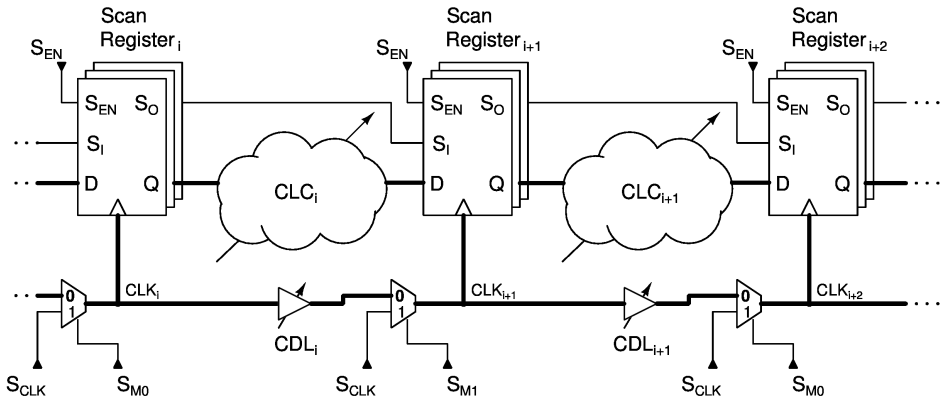

Figure 2.8 Multi-stage self-timed structure configured for regular operation.

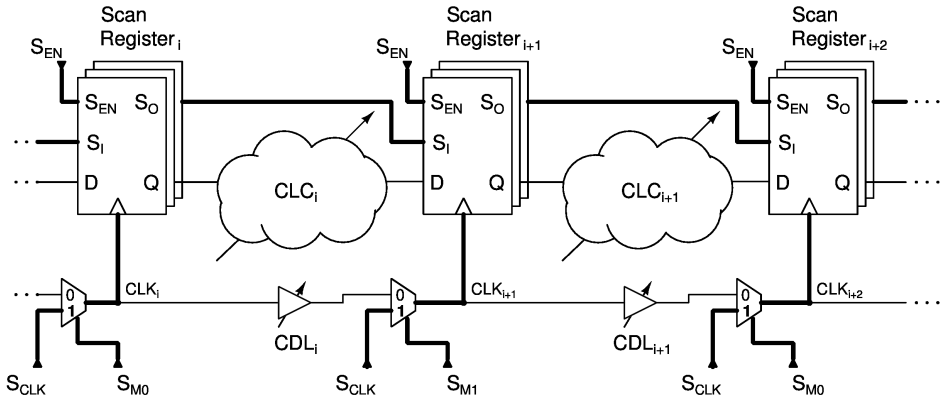

Figure 2.9 Multi-stage self-timed structure configured for shifting the test vectors synchronously.

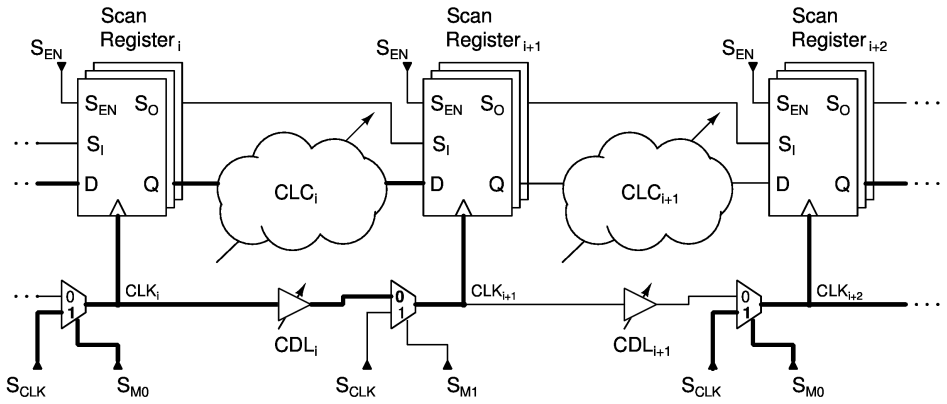

Figure 2.10 Multi-stage self-timed structure configured for at-speed launch and capture.

flops and is used to chose between the data input (D) or the scan input (S_I), as in regular synchronous designs [27]. Thus, no major timing impacts are expected with the insertion

of those multiplexers. Nonetheless, if needed, the delay line can be adjusted to negate the effects of any delay added by the multiplexers.

The second mode is the shift mode. This mode is used to scan-in or scan-out the test vectors. All the test control signals are set to 1 in this mode, and the shifting of the test vectors is done with a synchronous scan clock (S_{CLK}). Fig.2.9 shows the paths that are activated during the shift operation as thick lines.

The last mode of operation is the at-speed test mode. Before applying at-speed testing on this multi-stage self-timed structure, all the logic clouds and delay lines are preconfigured. Note also that the multi-stage self-timed structure is at-speed tested in an interleaved fashion, where one out of two consecutive stages will be tested at a time, thus requiring two steps to at-speed test the whole structure. This is done to simplify programming the ATPG tool and is designed to work with the AnARM LOC strategy (section 2.6.3).

To apply the at-speed test, the S_{EN} signal is set to 0, while the scan mode signals are set to opposite values. This setting would activate the paths highlighted in Fig. 2.10, where half of the registers would be launching the test vectors and the other half would be capturing the results. When the test vectors are ready, the S_{CLK} is pulsed only once. This pulse will launch the test vector into the preconfigured logic cloud, travel through the preconfigured delay line and capture the test result. When the scan mode signals are inverted, the register roles are switched and the second half of the structure is tested. This test is considered at-speed because, with respect to the logic clouds, the delay lines are configured to have the same speed as in the regular operation mode. Thus the data is captured at the speed of regular operation. Table 2.1 summarizes the different modes of operation and the settings of the test control signals.

2.6.3 AnARM LOC Strategy

An LOC based delay test strategy is adopted to apply the above at-speed test method. The LOC strategy removes the tight timing constraints from the test control signals, and simplifies programming the synchronous ATPG tool for the self-timed structure (section 2.6.4 elaborates more on the later point). Applying LOC on the AnARM self-timed architecture is done in three stages: shift-in, launch and capture, shift-out. As mentioned earlier, this procedure is repeated twice to test the entire multi-stage self-timed structure where the only difference between each repetition is the value of the scan mode signals.

Table 2.1 Summary of test modes

Mode	S_{EN}	S_{M0}	S_{M1}
Mission	0	0	0
Shift	1	1	1
At-Speed test	0	1 (0)	0 (1)

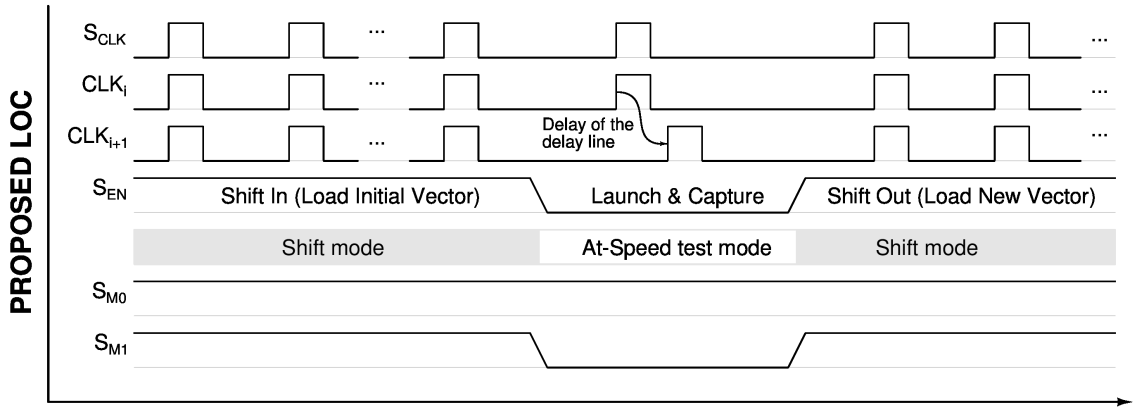

Figure 2.11 Timing diagram of the AnARM LOC strategy

In Fig. 2.11, the timing diagram of the AnARM LOC strategy is illustrated. The test starts by setting the system to shift mode (as described in Table 2.1). The initial test vector is shifted into the scan chains by pulsing the scan clock (S_{CLK}). Once the test vector is completely loaded, the system is switched to the at-speed test mode. Enough time is given for the test control signals to stabilize in order to ensure correct operation.

There is an important difference between a conventional synchronous LOC and the AnARM LOC. The fact that the capture pulse is internally generated in the AnARM self-timed structure means that, in the AnARM LOC, the S_{CLK} will only be pulsed once. This pulse travels to CLK_i, where it launches the data into the logic cloud and then travels through the delay line. When it arrives at CLK_{i+1}, it triggers the capture of the test results. Since the delay line is preconfigured to match the logic under test, this delay test happens at-speed. Next, the system is set again to the shift mode to scan-out the test results and scan-in the next test vector.

2.6.4 Pattern Generation Considerations

In order to correctly generate the patterns with conventional ATPG tools that target synchronous designs for the AnARM self-timed structure, while avoiding invalid test results, the following two issues must be addressed with regards to the use of a delay line to apply at-speed testing. The first issue arises from the fact that the capturing registers are considered to be in a different time-domain as compared to the launching registers, due to the presence of delay lines. This leads to the potential of having unexpected values in the launching registers at the end of the test. This issue is solved by masking the launching registers so that the results are only read on the capturing registers. The second issue arises from the need to write time-plates for the synchronous pattern generation tool in order to correctly describe the LOC timing. As seen in Fig. 2.11, the S_{CLK} does not pulse during the capture cycle, moreover, none of the test control signals change. Thus, for LOC, the time-plate for describing the capture cycle is a dummy time-plate that is used to stall the input signals until the pulse internally captures the data. Notice that since none of the signals in LOC changes between the launch and capture cycles, the use of synchronous ATPG tools for pattern generation is simplified. If an LOS approach was to be used, an additional scan enable signal would be necessary for a separate control of the launching and capturing registers. Thanks to the interleaved scheme, no flip-flop would have to switch from a launching ($S_{EN} = 1$) to a capturing ($S_{EN} = 0$) mode between the two clock pulses. Thus, the S_{EN} signals would not be critical, as they are for regular synchronous designs. In addition to the fact that it does not require an extra S_{EN} signal, the LOC scheme benefits from a greater ATPG tool flexibility, provided by the so-called "named capture procedure" [71].

2.7 Contributions and Publication Related to This Work

In light of the presented introduction and literature review, we list here the main contributions of the work presented in this dissertation:

1. The development of a new SDD test model and a flexible SDD test quality metric that can take advantage of the variable speed CDL in the Octasic structure.
2. The development of a method of assessing SDD test quality using the proposed metric under multiple PVT points.
3. The development of an optimized SDD for the AnARM structure test based on the proposed SDD test model and metric.
4. The verification and analysis of the previously presented at-speed test method of the AnARM. This is done through the application of the test on the AnARM chips and

confirmation that the inserted test structure and proposed LOC work and are effective in catching delay defects.

Note that the contribution of this work directly applies or can be expanded to regular synchronous systems in many parts. For that reason, many of the coming chapters will not specifically target the AnARM.

As of the date of writing this dissertation, several papers related to this work have been published or submitted. The following is a summary of these papers.

- "WeSPer: A Flexible Small Delay Defect Quality Metric" [66], published in VTS (conference) 2016
- "Exploiting Built-In Delay Lines for Applying Launch-on-Capture At-Speed Testing on Self-Timed Circuits" [72], published in VTS (conference) 2018
- "Multi-PVT-Point Analysis and Comparison of Recent Small-Delay Defect Quality Metrics", submitted to JETTA (journal) March 2019
- "Optimization of Small-Delay Defects Test Quality By Clock Speed Selection and Proper Masking Based on the Weighted Slack Percentage", submitted to TVLSI (journal) June 2019

Note that the contents of these papers form a large part of this dissertation. However, the reused parts of these papers have been rearranged, reformatted and/or expanded for the coherency and consistency of this dissertation. In addition, the contents of this dissertation is more detailed than the papers. It is difficult to map different sections of this work to these papers. However, the ideas in chapter 3 were presented in the VTS 2016 and JETTA 2019 papers, and the ideas in chapters 4 and 5 were presented in the VTS 2018 and TVLSI 2019 papers.

CHAPTER 3 THE IDEAL SDD TEST MODEL AND THE WEIGHTED SLACK PERCENTAGE

As the feature size of transistors gets smaller and smaller, the fabrication process becomes more complex and more difficult to control. This increases the amount of uncertainty in the delays of CMOS circuits, and with it, the task of formulating a perfect SDD test quality metric becomes almost impossible. As chapter 1 and 2 discussed, researchers in the SDD test domain have tried to formulate many kinds of metrics to address different aspects of delay testing. Many generic SDD test quality metrics have been developed in hopes of having a better representation of the test quality. The current non-statistical metrics are easy to calculate with the basic delay information that is available in any PDK. However, if an accurate delay defect distribution is known, using a statistical metric to estimate the quality of an SDD test is a better choice. In the literature, statistical metrics used a delay defect distribution that is not included in standard PDKs and that is not a function of voltage or temperature.

Generic SDD test quality metrics lack an ability to include aspects of the test other than path delays (or slacks) in their calculation. For example, path susceptibility to hazards, or delay induced due to cross-talk or voltage supply drooping cannot be integrated into those metrics. Also, if the goals of the test changes between including or excluding reliability defects in a test, for one reason or another, these metrics do not have the flexibility to include such a decision into the computation.

The self-timed AnARM structure, as have been shown before, has the ability to adaptively change the speed of the test while applying the test patterns. This feature is only useful in the context of FAST, where the test slack is minimized to minimize the chances of having test escapes. In this work, we want to exploit the adaptive speed of the AnARM self-timed structure to build a high-quality delay test. Our focus is on catching effective SDDs that can escape standard TDF based delay testing.

For all the reasons mentioned above, we propose a flexible metric that is more compatible with FAST and can be adapted to different SDD test methods according to the information available at hand. This chapter will introduce the formulation of the proposed metric that is built under our *Ideal SDD Test Model*, then discuss the simulation flow for multi-PVT-point analysis and present part of a very large set of results. The results and related discussion

will compare the proposed metric with some of the other SDD test quality metrics in terms of sensitivity to PVT changes, and propose how to estimate the test quality over a large set of PVT points. It will also discuss the significance of adding a delay defect distribution in the proposed metric. Note that, since the result set is large, many of the undiscussed results will be added in the appendix of this dissertation.

3.1 The Ideal SDD Test Model

To formulate the proposed metric, we start by defining the notion of a hypothetical *ideal SDD test*. Under the TDF model, let us assume that:

1. Path delay information can be exactly known (through a fictitious perfect simulations, for example).
2. CMOS circuit delays (i.e. path delays) do not vary as a function of any other factor (no process variations, cross-talk, etc.).
3. The longest true path through each fault site is known.
4. The combination of the test clock and applied pattern can test exactly the size of the smallest effective delay defect (SEDD) for each fault.

If a delay test is applied in an environment where these assumptions hold, we call it an *ideal SDD test*. Moreover, we define the size of the SEDD as the smallest possible slack, for each fault, under normal operating conditions. In other words, the SEDD size is the difference between the system clock period and the longest true path through a given fault site ($T_{sys} - PD_{LA_i}$).

Fig. 3.1 illustrates the concept of the ideal SDD test on an example circuit (Fig. 3.1(a)) with the delay of the annotated paths shown in Fig. 3.1(b). For the marked fault, the ideal test would be achieved if this fault is activated (tested) through the longest path that starts from I_4, and is observed at O_2, with a test clock period of $T_{test} = T_{sys}$. If, for any reason, the ATPG tool did not sensitize the longest path, the other options would be to test the marked fault through either path A or B. Since the delay of these paths are different, the ideal test can still be achieved if the test clock period is appropriately adapted to detect exactly the SEDD. In the example of Fig. 3.1, this is shown by setting the test clock period to T_{idealA} or T_{idealB} respectively. Notice that if the applied pattern activated both path A and path B, and if the test was to select path B and a test clock with a period of T_{idealB}, then the result should be observed at O_1, whereas O_2 should be masked because it has a negative slack. Thus in general, when applying a test pattern with a certain test clock speed, any endpoint that observes a negative slack should be masked to avoid false failures.

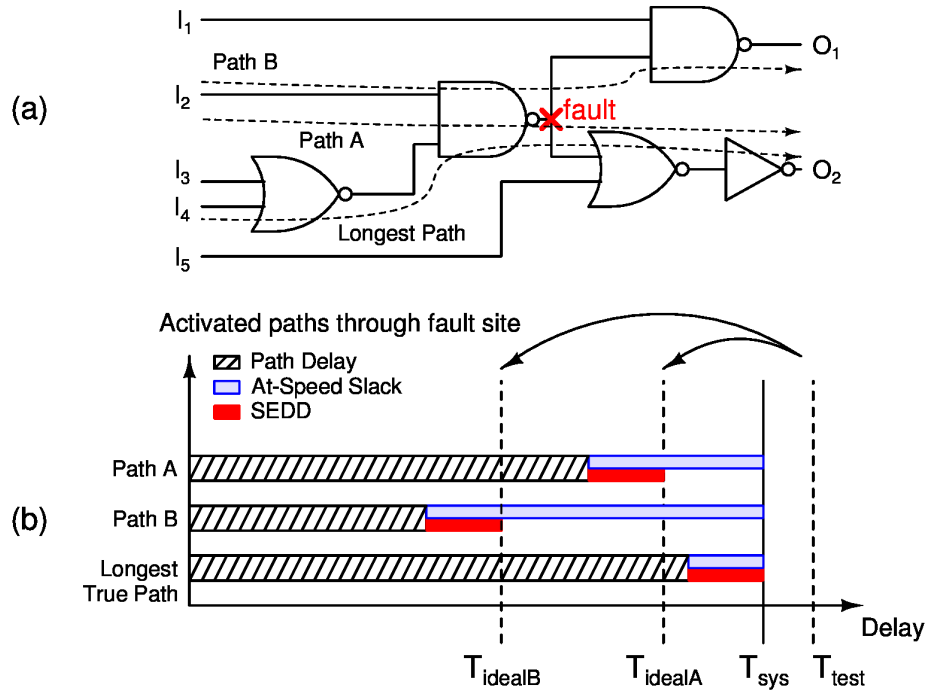

Figure 3.1 Example circuit with selected paths shown in (a) along with a path delay diagram shown in (b). In (b) the path delay diagram shows the smallest effective delay defect (SEDD) size of the marked fault and slack of each path along with their ideal test clock period

Finally, the idea of the proposed metric revolves around estimating the difference between the applied delay test and the proposed hypothetical ideal SDD test model. This is done by comparing the size of the minimum tested slack to the size of the SEDD, as well as representing any nonideality in a real delay test by a number (a percentage multiplier).

3.2 The Weighted Slack Percentage (WeSPer)

The proposed metric is called the *Weighted Slack Percentage (WeSPer)*. As the name implies, this metric is based on path slack ratios weighted by confidence level (CL) multipliers that are added according to information availability. WeSPer is calculated as follows:

$$WeSPer = \frac{1}{N}\sum_{i=1}^{N} f_i \, CL_i \times 100\% \tag{3.1a}$$

$$f_i = \frac{T_{sys} - PD_{LT_i}}{T_{test} - PD_{BA_i}} \quad \text{for} \quad T_{test} > PD_{BA_i} \tag{3.1b}$$

where N is the number of faults in the CUT, T_{sys} is the system clock period, T_{test} is the test clock period and, as defined earlier, PD_{LT_i} is the longest true path through a fault

site. PD_{BA_i} is the best activated path through a fault site and CL_i is the confidence level of testing through a fault site. The confidence level concept will be explained later in this section.

Basically, WeSPer is the percentage of the size of the minimum tested defect ($T_{test} - PD_{BA_i}$) in comparison to the size of the SEDD ($T_{sys} - PD_{LT_i}$) weighted by the confidence level of the test (CL). The slack terms that are used in WeSPer are very similar to what is defined in SDDC and SDQL with one key difference; in WeSPer, the tested slack is selected based on the best activated path rather than the longest. The best activated path is the path that maximizes the term $f_i\, CL_i$. In the reviewed metrics (in section 2.5), the best activated path is the longest one because it maximizes the metric. This is not necessarily true in WeSPer, since it depends on how the CL multipliers are calculated. Also, notice that f_i is defined only for positive slacks ($T_{test} > PD_{BA_i}$). Any endpoint that has a negative slack should be masked, as it is an invalid observation point for that pattern.

The CL_i are weighting factors that are calculated by equations and added by the test engineer according to available information. This term gives a great deal of flexibility to the proposed metric. The equation of the CL should be a weighting factor that rewards what the test engineer considers good and penalizes what he/she considers bad for the test. For example, in the case of hazards, a CL can be formulated to penalize test patterns that create hazards at endpoints. Another example is related to the availability of a delay defect distribution. If such a distribution is available for the used process technology, a CL can be added to give importance to testing certain sizes of defects over others. When multiple confidence level multipliers are defined, CL_i is formulated as follows:

$$CL_i = \begin{cases} \prod_{k=1}^{M} C_k & \text{for} \quad M > 0 \\ 1 & \text{for} \quad M = 0 \end{cases} \tag{3.2}$$

where $0 \leq C_k \leq 1$ is one type of confidence level multipliers and M is the total number of defined confidence level multipliers. If no CL multiplier is defined, CL is set to 1 and WeSPer can still be calculated. Thus, one can evaluate the SDD test quality while considering multiple factors that influence the test quality, rather than looking at disjointed quality indicators.

3.2.1 WeSPer for FAST

In this subsection, we will describe a CL multiplier for the case of overtesting in FAST. In this context, overtesting is defined as the case when the combination of the test pattern applied and test clock speed used test a delay defect size ($T_{test} - PD_{BA}$) that is less than the size of the SEDD ($T_{sys} - PD_{LT}$). In other words, a reliability defect. Based on this definition, and the fact that $PD_{LT} \geq PD_{BA}$, overtesting can only occur when applying a FAST ($T_{test} < T_{sys}$).

Most metrics are not designed to measure the quality of an SDD test in the case of overtesting, and when they do, they assign a perfect score for that test case. This is understandable since FAST is mainly used as a reliability test, where the test is designed to catch the smallest delay defect possible. Overtesting might be good as a reliability check, but if the goal is to catch effective SDDs, then it should be avoided since it tests for delay defects that cannot cause failure under normal operating conditions, and consequently, it could falsely label a good chip as defective. For that reason, a CL multiplier that penalizes overtested faults was added to WeSPer. This CL multiplier (denoted by CL_{OT}) is defined as follows:

$$CL_{OT_i} = \begin{cases} 1 & \text{if} \quad f_i \leq 1 \\ (1/f_i)^2 & \text{if} \quad f_i > 1 \end{cases} \tag{3.3}$$

where f_i is the same term defined in (3.1b). The term CL_{OT_i} was formulated to penalize overtesting with the same intensity as undertesting (testing SEDD with larger slacks). Notice that, when calculating WeSPer with CL_{OT}, the best activated path is the path with the maximum delay that does not cause overtesting.

Moreover, to measure how much overtesting occurred when using a faster-than-at-speed clock, we define a debugging metric named the *Total Overtesting Percentage (TOPer)*. TOPer is calculated as follows:

$$TOPer = \frac{1}{N} \sum_{i}^{N} OPer_i \times 100\% \tag{3.4a}$$

$$OPer_i = \begin{cases} 0 & \text{if} \quad f_i \leq 1 \\ 1 - (1/f_i) & \text{if} \quad f_i > 1 \end{cases} \tag{3.4b}$$

where f_i is the same term defined in (3.1b).

3.2.2 Statistical WeSPer

WeSPer has so far been defined as a non-statistical metric; simple to compute and does not require a delay defect distribution. However, if an accurate delay defect distribution is available, it is possible to add a statistical CL multiplier to WeSPer. The purpose of this statistical CL multiplier is to add proper weight to the size of the test escape window (i.e. the difference between the tested delay defect size and the SEDD size). Following the model presented in Eq. (3.2), the statistical CL multiplier is formulated as follows:

$$CL_{ST_i} = \begin{cases} 1 - \frac{\int_{SEDD}^{S_{test}} F(s)\,ds}{\int_0^{\infty} F(s)\,ds} & \text{if} \quad f_i \leq 1 \\ 1 - \frac{\int_{S_{test}}^{SEDD} F(s)\,ds}{\int_0^{\infty} F(s)\,ds} & \text{if} \quad f_i > 1 \end{cases} \tag{3.5}$$

where f_i is the same term defined in (3.1b), $F(s)$ is a delay defect distribution expressed as a function of the defect size s, $SEDD$ is the smallest effective delay defect ($T_{sys} - PD_{LT}$) and S_{test} is the size of the tested delay defect ($T_{test} - PD_{BA}$). The CL_{ST} equation is a measure of the probability of missing an effective delay defect due to the difference between the tested defect size and the $SEDD$ size (test escape window size). In the case of overtesting ($f_i > 1$), CL_{ST} would penalize the test based on the probability of catching a delay defect that is smaller than the $SEDD$. In the coming sections, to indicate when WeSPer is using this statistical CL, the term WeSPer$_{ST}$ will be used.

3.3 Multi PVT Point Metric Computation Flow

During this project, we developed a metric computation framework that takes a set of well known industrial tools and uses them to extract information needed to compute WeSPer and other metrics. Some of the information required are not provided directly by the tool. A work around was done with proper scripts in TCL to extract the needed information and feed it to python calculator that computes all metric. The framework initially was designed for a single PVT calculation and later was grown to be multi-PVT-point analysis tool of SDD metrics.

To better understand the coming results and the challenges that were faced to obtain them, this section will discuss the computation flow and tools framework used to produce

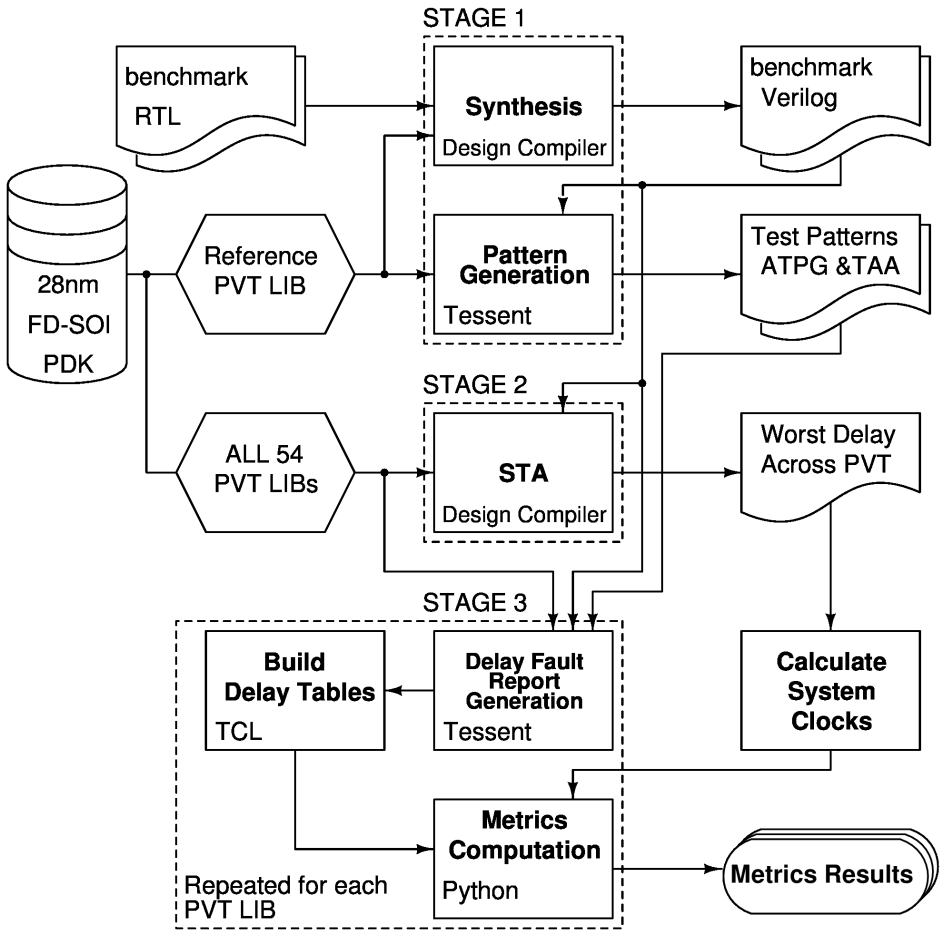

Figure 3.2 The computation flow for metric computation for all 54 PVT points.

these results. In the multi-PVT-point analysis, we look at how different metric results change with PVT point changes. The goal is not to maximize the metrics by optimizing the test for each PVT point, but rather to look at the effects of PVT point changes on the SDD test quality as reported by the selected metrics. The presented flow is analogous to synthesizing a digital design at a single PVT point, but the timing constraints are checked over all PVT points. Thus, in this flow, the benchmark circuits synthesis and test patterns generation are done under an arbitrarily selected reference PVT point. Then SDD test quality metrics are computed for all available PVT points. By doing so, we guarantee that the same circuit structure and test patterns are used for all the PVT points, and that the only variable that affects the circuit path delays is the PVT point variation. Moreover, for each benchmark circuit, a single system clock is chosen based on the critical path of the slowest PVT point. This is automatically done in the flow by finding the critical path of each circuit under all PVT points and determining the longest one. This work uses a version of the 28nm FDSOI PDK by STMicroelectronics that has 54 different PVT points.

Fig. 3.2 shows the computation flow diagram of the metrics for the selected benchmark circuits for all available PVT points. The first stage in the flow is to generate the Verilog netlist (using Synopsys Design Compiler [73]) and test patterns (using Mentor Tessent [74]) for all benchmark circuits for the reference PVT point. Stage 2 uses the circuits netlist and computes the worst critical path delay for each circuit under all PVT points. Using that delay, the system clock speed for each benchmark circuit is computed. Those two stages are relatively fast to process.

Stage 3 involves a more tedious computation. In order to compute all the metrics, we need to find the true path delay, for both the longest path and the activated (tested) path, for each fault. A whole field of research is dedicated to finding the longest path for delay faults [75–77] and reviewing it is outside the scope of this work. Moreover, the proposed ideal test model of WeSPer requires finding the best activated path rather than the longest path due to the use of CL multipliers (as explained in section 3.2). Since the calculation of WeSPer requires finding the best activated path for each fault, the extraction of such information from Tessent shell is not straightforward. By default, Tessent shell only reports the max activated path delay for each fault (for a whole set of patterns and for all possible endpoints), whereas WeSPer requires finding the activated true path delay information (for rising and falling transitions) through each fault site to every possible endpoint under each pattern. The delay information for each fault under each pattern to every possible endpoint is extracted from the tool by applying the pattern vectors one at a time while masking endpoints appropriately. The delay information is collected from Tessent shell fault reports when the timing engine is activated. These fault delay information are the ones that Tessent uses to compute the DTC. Since Tessent is not optimized to extract the delay information for each pattern and every endpoint separately, the generation of these fault reports is very time consuming. In the literature, there are other tools that are much more optimized and capable of extracting such information in a fraction of the time that we observed [76], however, they are not available to us.

From the extracted delay reports generated with Tessent shell, a TCL script builds a set of two delay tables. One is a table of the longest true path delay for each fault and the other is a table of the tested (activated) path delay for each fault under each pattern for each endpoint (as in Fig. 3.3). Those two tables, along with the previously calculated system clocks, are used by a Python script to compute all the SDD test quality metrics that are discussed in the next section. Note that stage 3 of the computation flow is repeated for each PVT point calculation until the metric results for all PVT points are collected. Due to disk

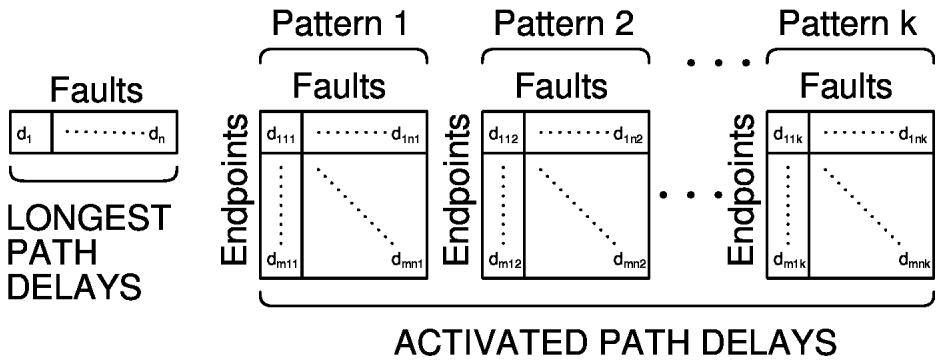

Figure 3.3 Path delay tables

space limitation, delay tables are deleted after the metrics are computed for each PVT.

3.4 Simulation Results and Discussion

The following results were obtained from the computation flow presented in section 3.3 under the 28nm FD-SOI technology from STMicroelectronics on a set of ISCAS-85, 74X-Series and ITC99 benchmark circuits. Fig. 3.2 depicted the computation flow and indicated the tools used for each step. As mentioned earlier, the version of the 28nm FD-SOI PDK used contains 54 PVT points. The reference PVT point used in the flow is selected to be at the typical process point, operating at 0.9V and 25°C. The selected reference PVT point is somewhat in the center of the PVT space of this technology. This helps to show the effects of faster and slower PVT points on the presented results. For all the coming results, to compute the metrics:

- The system clock period is chosen to be 25% slower than the delay of the worst critical path of the benchmark circuit under the slowest PVT point.
- The metrics are computed for three test clock speeds: slow, at-speed and fast. In reference to the system clock period, the slow, at-speed and fast test clock periods are 1.1, 1 and 0.9 times the system clock period, respectively.
- The results are computed for two sets of patterns: conventional TDF patterns, and timing-aware TDF patterns. For brevity, these pattern sets will be referred to as ATPG patterns and TAA patterns, respectively.

Table 3.1 shows the characteristics of the benchmark circuits used in the calculation. For each circuit, this table lists the number of faults, endpoints and patterns in each pattern set for each circuit. The sixth column in the table lists the average percentage of increase in the activated path delay (PD_A) when applying the TAA pattern set, and the last column

Table 3.1 Benchmark circuit characteristics

Circuit	# of Faults	# of Endpoints	# of patterns ATPG	# of patterns TAA	TAA Average PD_A increase (%)	TDFC (%)
c17	34	2	7	8	4.21	100
b06	184	9	28	31	3.77	89.13
x74283	202	5	30	36	2.02	100
x74181	354	8	46	62	3.43	100
b08	532	21	81	94	2.19	96.05
c432	812	7	85	85	23.34	98.77
c499	934	32	100	124	2.80	99.14
b07	1032	44	109	116	8.89	96.80
c880	1556	26	85	99	14.78	100
c1908	2048	25	161	190	10.24	99.66
c2670	3268	140	125	175	7.73	97.71
c3540	4862	22	231	250	3.22	97.90
c5315	8028	123	144	179	11.02	99.41
c7552	10298	108	191	228	3.75	98.72

lists the TDFC for each circuit. The TDFC value is the same for both the ATPG and TAA pattern sets.

In this section we will start with showing the results for a single PVT point (the reference point) and compare WeSPer with the presented metrics in section 2.5. The SDFC is the only metric excluded from this comparison because it needs path delay distribution, which is not supported by our flow at the moment. The results are then expanded to compare the SDD test quality metrics on the full results of 54 PVT points and discuss three more aspects: metric sensitivity, predictability and worst-case scenario. Due to the huge number of results obtained, it is impossible to list all the results in few pages. This section will also include the complete set of results for one circuit, one pattern set and three test clock speeds and follow with statistical information that summarizes the full set of obtained results. The full set of results are then listed in the appendix for reference.

Note that in the coming results, the calculation of the statistical metrics (SDQL and SDDC) requires a delay defect distribution. Unfortunately, the delay defect distribution for the 28nm technology is not available to us, and adapting this distribution from other technologies is not trivial. Therefore, for the sake of generating results for the statistical metrics, we used the same delay defect distribution used in [20, 62] as an approximation. This delay defect

distribution is expressed as follows:

$$F(s) = 1.58 \times 10^{-3} \times e^{-2.1\,s} + 4.94 \times 10^{-6}(DPM) \tag{3.6}$$

where s is the delay defect size in nanoseconds.

In the coming multi-PVT simulation results, the TDFC will be reported in the results as it serves as the theoretical upper limit for the selected metrics. Whenever any of the reported metrics exceeds the TDFC, that metric is reporting an invalid value. In addition, we introduce a comparison measure called the *mean slack difference (MSD)*. The MSD is the average difference between the minimum defect size detected by the test and the SEDD. It is calculated by averaging the absolute difference between the tested slack and the minimum slack for each fault over the complete circuit. In terms of the clocks and path delay it is expressed as follows:

$$MSD = \frac{1}{N}\sum_{i=1}^{N} |(T_{sys} - PD_{LT_i}) - (T_{test} - PD_{A_i})| \tag{3.7}$$

where N is the number of faults in the circuit and the rest of the terms are as previously defined in section 2.5. In a qualitative terms, the MSD is considered as the average size of the test-escape (or overtest) window for a circuit measured in nanoseconds. The closer the MSD is to zero, the closer the test is to an ideal SDD test. The MSD will be used to better understand the relation between each of the computed metrics and the test-escape window.

3.4.1 Single PVT results

Table 3.2 shows metric computation results for a single PVT (typical process, 0.9V, 25°C). This is the same PVT point used as a reference for the multi-PVT simulations. In this table, each of the metrics are computed with 3 test clock speeds: slow, at-speed and fast. Note that since these results are part of the multi-PVT simulations, the system clock period here is set to be 25% larger than the longest path delay of the CUT under the slowest PVT point. The table also shows the results for both an ATPG pattern set and a TAA pattern set.

First, looking at how the test quality changes for each metric with respect to the test clock speed, it can be seen that all metrics report a change in the test quality when the test clock speed changes, except for the DTC. As an example, the DTC value remains at 94.31% for the c17 circuit, ATPG patterns, for all the 3 clock speeds (S, A, F). This is because the equation

Table 3.2 Metric results with the test clock speed indicated with respect to the system clock by (S) when it is 10% slower , (A) when it is at-speed and (F) when is 10% faster

	ATPG pattern results					TAA pattern results				
Circuits (Speed)	**DTC (%)**	**SDQL (DPM)**	**SDDC (%)**	**SDDC^Q (%)**	**WeSPer (%)**	**DTC (%)**	**SDQL (DPM)**	**SDDC (%)**	**SDDC^Q (%)**	**WeSPer (%)**
c17 (S)	94.31	945.39	93.37	17.62	88.69	98.84	883.28	93.80	19.31	89.36
c17 (A)	94.31	83.34	99.42	90.67	98.95	98.84	17.67	99.88	98.08	99.78
c17 (F)	94.31	0.00	**105.82**	**324.23**	93.25	98.84	0.00	**106.30**	**337.35**	93.61
b06 (S)	84.87	395.04	86.38	77.89	79.86	85.11	389.51	86.41	77.30	79.95
b06 (A)	84.87	9.48	89.06	81.60	88.88	85.11	9.11	89.07	81.88	88.89
b06 (F)	84.87	0.00	92.36	**587.46**	79.60	85.11	0.00	92.29	**606.19**	79.77
x74283 (S)	98.00	6115.64	87.12	15.20	89.40	99.18	6015.45	87.33	15.70	89.56
x74283 (A)	98.00	188.61	99.61	96.33	99.66	99.18	73.51	99.85	98.42	99.87
x74283 (F)	98.00	0.00	**113.96**	**339.50**	91.52	99.18	0.00	**114.23**	**343.17**	91.50
x74181 (S)	85.89	6415.81	79.20	11.26	88.05	89.51	6136.49	80.05	13.18	88.57
x74181 (A)	85.89	1177.11	96.28	75.05	97.90	89.51	808.23	97.40	80.92	98.55
x74181 (F)	85.89	0.00	**118.85**	**321.59**	94.98	89.51	0.00	**120.33**	**331.42**	94.91
b08 (S)	85.17	1720.18	93.52	**261.01**	86.39	86.92	1707.35	93.54	**250.71**	86.41
b08 (A)	85.17	91.02	95.92	77.05	95.50	86.92	78.06	95.94	79.92	95.57
b08 (F)	85.17	0.00	98.32	**1037.93**	87.66	86.92	0.00	98.34	**1023.86**	87.66
c432 (S)	61.14	3375.50	94.99	**293.66**	83.72	73.81	2968.82	95.44	**290.87**	85.29
c432 (A)	61.14	1220.49	97.41	42.39	92.78	73.81	810.59	97.86	58.60	94.71
c432 (F)	61.14	67.72	99.85	**666.20**	95.74	73.81	26.64	**100.32**	**701.41**	95.73
c499 (S)	85.63	9680.15	82.96	**787.95**	86.52	89.35	9248.19	83.67	**792.56**	87.13
c499 (A)	85.63	2116.19	95.67	78.07	96.57	89.35	1494.32	96.69	83.58	97.32
c499 (F)	85.63	187.04	**114.13**	**1290.63**	94.62	89.35	126.90	**115.62**	**1297.05**	94.46
b07 (S)	78.43	4133.85	94.19	**369.93**	86.89	80.61	4098.33	94.22	**360.47**	86.96
b07 (A)	78.43	308.85	96.61	66.40	96.00	80.61	273.31	96.63	69.50	96.09
b07 (F)	78.43	0.00	99.02	**1153.16**	88.85	80.61	0.00	99.04	**1159.02**	88.74
c880 (S)	85.53	5381.50	94.37	**5456.33**	88.36	90.28	5129.18	94.62	**5455.51**	88.91
c880 (A)	85.53	1120.59	98.85	75.83	97.88	90.28	774.05	99.21	83.05	98.55
c880 (F)	85.53	4.56	**105.12**	**6664.94**	94.48	90.28	0.00	**105.65**	**6680.21**	94.45
c1908 (S)	70.42	6715.22	95.52	**278.64**	85.26	76.92	6290.19	95.77	**277.76**	86.10
c1908 (A)	70.42	2250.02	98.28	53.91	94.68	76.92	1776.36	98.57	62.37	95.71
c1908 (F)	70.42	44.07	**101.47**	**665.47**	97.35	76.92	13.36	**101.84**	**683.27**	97.37
c2670 (S)	83.04	9345.36	93.40	**5092.57**	86.33	87.87	8853.82	93.62	**5092.07**	86.91
c2670 (A)	83.04	1788.85	96.89	73.22	95.68	87.87	1150.84	97.18	80.81	96.40
c2670 (F)	83.04	9.63	**101.50**	**5861.93**	92.50	87.87	3.03	**101.92**	**5877.59**	92.50
c3540 (S)	77.27	17124.02	94.83	**1710.86**	85.47	79.88	16654.17	94.92	**1710.17**	85.76
c3540 (A)	77.27	3801.81	97.22	63.64	94.84	79.88	3329.61	97.31	67.18	95.20
c3540 (F)	77.27	9.94	99.64	**2127.99**	94.59	79.88	7.96	99.72	**2135.71**	94.58
c5315 (S)	84.26	22894.44	96.52	**10918.71**	88.10	88.74	22056.99	96.61	**10914.02**	88.46
c5315 (A)	84.26	3389.83	98.99	73.88	97.56	88.74	2539.71	99.08	80.67	98.00
c5315 (F)	84.26	16.32	**101.53**	**12130.13**	93.52	88.74	1.03	**101.63**	**12148.34**	93.54
c7552 (S)	80.14	29636.12	95.66	**1517.12**	86.67	83.98	28598.14	95.75	**1515.52**	87.04
c7552 (A)	80.14	5744.00	98.13	67.10	96.18	83.98	4676.19	98.23	73.10	96.63
c7552 (F)	80.14	4.13	**100.72**	**1944.84**	94.35	83.98	1.60	**100.83**	**1958.43**	94.37

of the DTC does not consider the test clock speed in the equation. The DTC, however, does report an increase of quality when using the TAA pattern set. As an example, DTC goes from 94.31 to 98.84% for the c17 circuit. This is expected, since the TAA pattern set sensitize longer paths (as can be seen in table 3.1). For the SDQL, the results are in DPM. This makes it difficult to know which test did better across circuits because the number of faults in a circuit is different. Nonetheless, the SDQL shows an improvement in the test as the test clock speed increases, or when sensitizing longer paths (TAA pattern is used). Notice that in many cases, when the test clock is fast, the SDQL reports a zero. This can be seen, for example, in the c17 circuit (ATPG and TAA). As mentioned before, dealing with negative values due to faster-than-at-speed clock was never discussed by the authors of SDQL [20], so in our computation we limited it to zero. This makes the SDQL useless when using FAST. The SDDC, like the SDQL is a statistical metric, but is formulated into a percentage value. This makes it more obvious to say that, for example, the c17 delay test is better than the c499 one at the slow test clock with the ATPG pattern set. SDDC cannot be used directly to assess delay test with fast clocks as it sometimes reports numbers $> 100\%$ (emphasized in the table for clearness with an underline and bold text). An example of that case is in the computation of the c17 for both ATPG and TAA pattern sets. The same can be said about the non-statistical version. The $SDDC^Q$ reports illogical values in some cases when the test clock is not at-speed. Notice that The SDQL, SDDC and $SDDC^Q$ report an improvement in the test quality when using the TAA pattern set over the ATPG pattern set.

As mentioned before, when overtesting occurs in a test, the SDDC metric is split into two metrics, the SDDC_{DPM} and SDDC_{EFR}. The SDDC_{DPM} is the SDD test quality and is limited to 100%, and SDDC_{EFR} represents percentage of reliability defects testing. Note that the SDDC_{DPM} 100% limit comes from limiting the test score of each fault to a max of 1. The SDDC_{EFR} is calculated as $SDDC - SDDC_{DPM}$. As there is no upper bound for the SDDC when overtesting occurs, the SDDC_{EFR} also does not have an upper limit. This makes it difficult to understand the significance of the values reported by $SDDC_{EFR}$. One could argue that the SDDC_{EFR} represents the percentage of the applied test that targets reliability defects. However, theoretically, if SDDC is 250% and SDDC_{DPM} is 100%, then $SDDC_{EFR}$ is 150%. This means that 150% of the applied test targeted reliability defects?! Fortunately, unless the faster-than-at-speed test clock puts the test deeply into the activity region of the CUT, it is unlikely to have such values. The SDDC_{DPM} and SDDC_{EFR} results for the benchmark circuits are listed in table 3.3.

Table 3.3 Single PVT overtesting results

	ATPG pattern results					TAA pattern results				
Circuits	SDDC DPM (%)	SDDC EFR (%)	WeSPer (%)	TOPer (%)	Overtesting avoided (%)	SDDC DPM (%)	SDDC EFR (%)	WeSPer (%)	TOPer (%)	Overtesting avoided (%)
c17	100.00	5.82	93.25	6.64	5.88	100.00	6.30	93.61	6.29	2.94
b06	89.13	3.23	79.60	6.86	17.68	89.13	3.16	79.77	7.10	14.02
x74283	100.00	13.96	91.52	8.43	1.98	100.00	14.23	91.50	8.45	1.98
x74181	100.00	18.85	94.98	4.94	7.34	100.00	20.33	94.91	5.01	7.91
b08	96.05	2.27	87.66	8.11	4.89	96.05	2.29	87.66	7.93	7.63
c432	98.69	1.16	95.74	2.34	46.99	98.74	1.58	95.73	2.46	49.81
c499	98.84	15.29	94.62	3.82	22.76	98.94	16.68	94.46	4.10	22.88
b07	96.80	2.22	88.85	7.66	9.41	96.80	2.24	88.74	7.79	8.41
c880	100.00	5.13	94.48	5.40	18.05	100.00	5.65	94.45	5.44	16.26
c1908	99.63	1.84	97.35	1.91	59.01	99.65	2.19	97.37	1.96	59.38
c2670	97.70	3.80	92.50	5.05	12.47	97.70	4.22	92.50	5.08	13.31
c3540	97.90	1.74	94.59	3.18	36.08	97.90	1.82	94.58	3.19	36.17
c5315	99.41	2.12	93.52	5.83	11.66	99.40	2.23	93.54	5.82	12.01
c7552	98.72	2.00	94.35	4.27	21.74	98.71	2.12	94.37	4.26	22.11

WeSPer was formulated to better fit the needs of this work by taking care of the case of FAST and not requiring a delay defect distribution to be computed. In the results in table 3.2, WeSPer is computed with the overtesting CL multiplier. This means that overtesting in FAST is penalized. In the results in general, WeSPer does not saturate nor report an invalid value. The closer the test to the ideal SDD test model, the higher the reported value. For example, as the test speed increase from slow to at-speed, WeSPer reports an increase in the quality value. When a faster-than-at-speed clock is used, the test quality drops if overtesting starts to be significant (e.g. c5315). Else the quality improves (e.g. c1908). The percentage of overtesting that occurs in a test is evaluated by TOPer (table 3.3). Using longer sensitized paths can also increase overtesting and reduce the WeSPer value. This can be seen, although very slightly, when comparing the ATPG pattern results versus the TAA pattern results of the x74181 benchmark circuit.

Finally, to show that WeSPer selects the best activated path rather than the longest one, we list in table 3.3 the percentage of overtesting avoided during the computation. As previously mentioned at the begining of this chapter, the focus of this work is on catching effective SDDs and not reliability defects. Thus, overtesting is penalized and avoided whenever possible, and the best path is the one that maximizes WeSPer without overtesting the CUT. We are assuming here that overtested endpoints can be masked during the test. It is difficult to predict how much overtesting can be avoided in a test as it a function of the CUT structure and the set of paths activated by the test patterns.

In the next subsection we will present more metric comparison results but with respect to PVT point change. However, the SDQL and SDDC will be excluded from the coming results. This is because the SDQL is not good for comparison, since it is not normalized over the number of faults, and the $SDDC^Q$ can have illogical results ($\gg 100\%$) when the test clock is not at-speed with the system clock.

3.4.2 Multi-PVT Single Benchmark Analysis

Fig. 3.4 plots the results of the c5315 benchmark circuit with an ATPG pattern set versus each PVT point under the three test speed cases (slow, at-speed, fast). The Metrics (TDFC, DTC, SDDC, SDDC_{DPM} and WeSPer) are plotted with respect to the left Y-axis. This plot shows how each of the selected metrics changes with the change of PVT points. For clarity, the PVT points were ordered from slowest to fastest based on the maximum activated path delay. The plot shows that the DTC value does not change with the change of the test clock speed, and has a change of less than 1% across all PVT points. For the SDDC and SDDC_{DPM}, their values get better and closer to 100% when the test clock speed changes from slow to at-speed. However, when the test clock is faster than the system clock (i.e. FAST), the SDDC value is no longer valid ($>$ TDFC and $>$ 100%) and the SDDC_{DPM} value saturates in most PVT points. The SDDC and SDDC_{DPM} are most sensitive to PVT changes under the slow test clock (7.3% variation) and least sensitive under the fast test clock ($<$ 5% variation). On the other hand, WeSPer is the most sensitive metric to test clock speed change and PVT point change. As the test clock speed increases, WeSPer results improves if no overtesting happens. This is apparent when comparing the slow PVT points (e.g. at SS/0.6V/-40°C) across the test speed changes. At that PVT point, WeSPer value increases from 86% to 90% when the test clock speed increases from at-speed to fast. However, at faster PVT points, some faults start to be overtested and the penalization of the overtested points shows in the figure as an overall decrease in WeSPer value when moving to faster PVT points. In this figure, the WeSPer value for the fast test clock starts becoming lower than the value of the at-speed clock at the FF/0.7V/-40C PVT point. Similar to the SDDC metric, WeSPer is most sensitive to changes when the test clock speed is slow (15.6% variation for c5315), whereas it is less sensitive when using the fast clock speed (5.5% variation for c5315) due to the penalization of overtesting.

Fig. 3.4 also shows the value of the MSDs (labeled *Slack Diff* and *WeSPer Slack Diff*) on the right Y-axis. There are two MSD values shown: the regular slack difference that is used with other metrics by considering the longest tested path delay through a fault site, and the

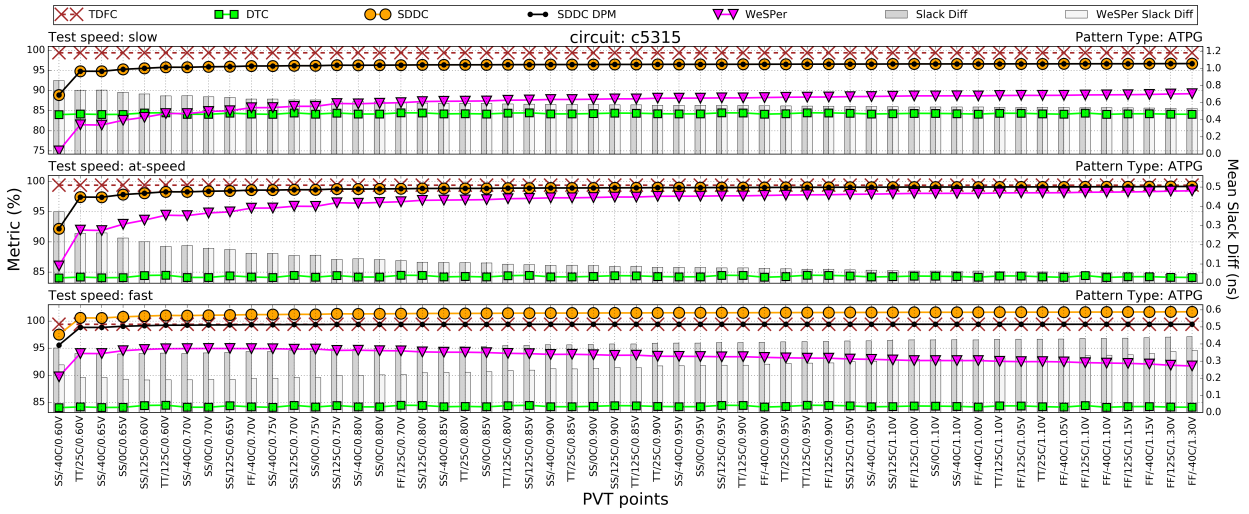

Figure 3.4 Metric results for all available PVT points for the c5315. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

WeSPer slack difference that uses the delay of the best tested (activated) path instead of the longest one (as discussed in section 3.1). The regular slack difference and the WeSPer slack difference are only different when overtesting occurs. This is because the best activated paths, in the case of a fast clock, are often not the longest ones, but rather the ones that result in a test slack closest to the SEDD size without overtesting. As a consequence, in the fast test subplot of the c5315, the MSD measured in WeSPer is lower than the regular one used by the other metrics in all PVT points.

Another interesting result can be seen by looking at the MSD results in the at-speed subplot of Fig. 3.4. Given that both the test clock and system clock speeds are equal and constant across PVT points in that subplot, changes in the MSD value (average test-escape window size) are mainly due to the difference between the longest path delay and the activated path delay for each fault. It can be seen here that, faster PVT points (e.g. a higher voltage point) produce smaller test-escape windows. This can be explained by the fact that faster PVT points produce tighter path delay distributions than slower ones [7].

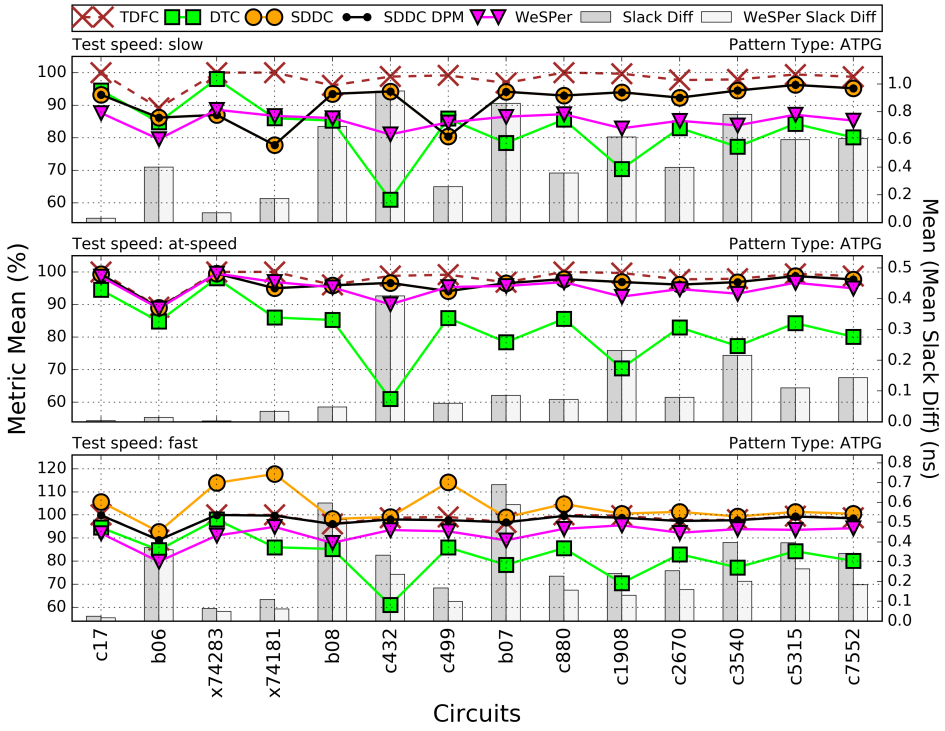

Figure 3.5 Metrics results summary for a test applied on the selected benchmark circuit set using a regular ATPG pattern set for three test clock speeds: slow (top), at-speed (middle) and fast (bottom). The mean metric values across PVT points are plotted against the left Y-axis, whereas the right Y-axis plots the mean of the mean slack difference.

3.4.3 Benchmark Circuits Results Summary

The conclusions from the results of the c5315 are also valid for the rest of the benchmark circuits and they reinforce the conclusions from subsection 3.4.1. To summarize the results for all other benchmark circuits, the mean values of the results for each circuit across all PVT points will be shown[1]. Fig. 3.5 shows the mean of each metric and MSD results for each benchmark circuit across PVT points for a conventional ATPG pattern set. In this figure, the results for all the metrics are plotted with respect to the left Y-axis, whereas the MSDs are plotted with respect to the right Y-axis. This figure shows how each metric reacts to the change of test clock speed for each of the benchmark circuits. The mean value of the DTC does not change with the test clock speed, and thus does not show the benefit of using a better test clock speed. On the other hand, all the other metrics show the advantage of using at-speed test clock over a slower test clock.

[1]The expanded PVT results of metric calculations for each of the benchmark circuits are included in appendix A.

Table 3.4 TAA pattern set results for metrics with respect to ATPG pattern set result. Values represent the (%) of improvement.

		Slow test		At-speed test		Fast test		
Circuit	**DTC**	**SDDC**	**WeSPer**	**SDDC**	**WeSPer**	**SDDC**	**SDDC$_{DPM}$**	**WeSPer**
c17	4.53	0.56	1.01	0.56	1.16	0.56	0.10	0.32
b06	0.28	0.04	0.10	0.00	0.01	-0.07	0.00	0.21
x74283	1.17	0.30	0.27	0.30	0.31	0.30	0.02	0.05
x74181	4.19	1.49	0.86	1.60	0.99	1.70	0.26	0.12
b08	2.02	0.03	0.04	0.03	0.11	0.03	0.00	-0.10
c432	20.72	0.64	2.71	0.67	3.08	0.77	0.36	1.14
c499	4.21	1.30	1.00	1.57	1.16	1.85	0.52	0.28
b07	2.79	0.03	0.10	0.03	0.13	0.03	0.00	-0.10
c880	5.53	0.46	0.87	0.65	0.99	0.96	0.26	0.25
c1908	9.20	0.40	1.40	0.51	1.59	0.75	0.35	0.56
c2670	5.82	0.42	0.97	0.60	1.11	0.90	0.19	0.14
c3540	3.38	0.12	0.49	0.12	0.55	0.13	0.05	0.19
c5315	5.32	0.13	0.57	0.14	0.65	0.17	0.03	0.15
c7552	4.79	0.14	0.60	0.15	0.67	0.19	0.04	0.14
Average	5.28	0.43	0.79	0.50	0.89	0.59	0.16	0.24

The case of faster-than-at-speed test clock is unique. The concept of penalizing overtesting was not introduced in metrics before WeSPer. Hence, in Fig. 3.5, SDDC and WeSPer handle that case differently. The SDDC and SDDC$_{DPM}$ values are exactly the same when no overtesting happens. However, in the case of a fast clock, the SDDC value goes above the TDFC value (the upper limit) which could be confusing or misleading. The authors of SDDC would refer to the SDDC$_{DPM}$ and SDDC$_{EFR}$ in this case. It can be seen that the SDDC$_{DPM}$ saturates in all the selected circuits indicating a perfect SDD test, and the comparison of the quality of the test is left for the SDDC$_{EFR}$ (results shown later in this section). In contrast, because WeSPer penalizes overtesting, it does not saturate. This makes it possible to indicate which test is better with a single metric. We should note that because, SDDC and WeSPer are built on different views on overtesting. The choice of which metric is better to use in this case depends on whether the goal of the test is to catch effective SDDs that escape TDF based delay testing or to test for reliability defects as well.

To summarize the results of the TAA pattern set, table 3.4 lists the percentage of improvement in the results when using TAA pattern set over the ATPG pattern set. The DTC

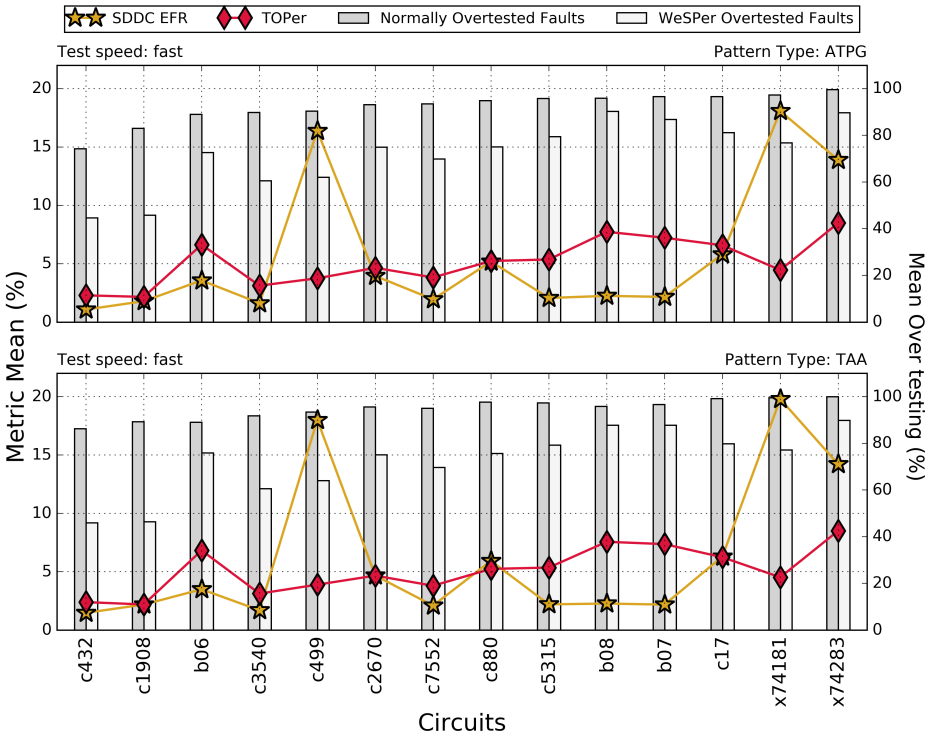

Figure 3.6 Overtesting results summary. The lines show the mean value for the overtesting metrics across all PVT points with respect to the left Y-axis. The right Y-axis along with the bars show the mean percentage of overtested faults with respect to the total number of faults for WeSPer and other metrics across all PVT points.

is the most affected metric to the change of pattern sets. The improvement is in the DTC reaches up to 20.72%. Whereas the SDDC and WeSPer improvement is limited ($< 4\%$). When comparing this improvement to the improvement gained from using an at-speed clock versus a slow clock, it suggests that using a better clock is better for the SDD test quality than using a TAA pattern set over an ATPG one. Notice that on average, the improvement from using a TAA pattern set increases with the increased clock speed, except in the case of WeSPer and faster-than-at-speed clocks. In that case, overtesting increases because the TAA pattern set sensitizes on average longer paths than the ATPG, and thus WeSPer improvement decreases. In the b07 and b08, for example, the WeSPer value is slightly less than the ATPG case (negative value).

To analyze how overtesting is affecting the results in Fig. 3.5 in the FAST case, Fig. 3.6 plots, with respect to the left Y-axis, the mean TOPer and mean SDDC_{EFR} across PVT points for each benchmark circuit and pattern set type. The right Y-axis of the figure reports the mean percentage of overtesting across all PVT points. It represents the percentage of

overtested faults to the total number of faults in a circuit. The two bars in Fig. 3.6 indicate the overtesting percentage when calculating other metrics and when calculating WeSPer. Remember that WeSPer looks for the best activated path and assumes that masking can be applied to overtested endpoints when possible. This leads to having a lower overtested fault percentage for all the benchmark circuits.

TOPer is used as a debugging metric to measure overtesting and explain the drop in WeSPer, if needed. In Fig. 3.6, a correlation can be seen between TOPer and the percentage of overtested faults. In contrast, SDDC_{EFR} is a complementary metric to SDDC_{DPM}, that measures the percentage of reliability testing in a test. It cannot be used as an overtesting measure as it does not correlate to the percentage of overtested endpoint.

3.4.4 Metric Sensitivity

It is obvious that metrics need to be sensitive to any slight changes in a circuit that can affect the quality of the test. As explained earlier in section 3.4, the MSD represents the average size of the test-escape or overtest window. As seen in Fig. 3.4, the size of that window changes with PVT point variations, and thus, if a metric is accurate, its value should change accordingly. However, the sensitivity of a metric is considered a sign of accuracy only if it reacts to the window size change in a correct way. In this case, we expect metrics to be inversely correlated to the test-escape window size (i.e. MSD).

In section 3.4.2, we briefly discussed the sensitivity of SDD test quality metrics in the case of the c5315 circuit. The sensitivity of a metric is calculated as the percentage of change in the metric value across all PVT points. In Fig. 3.7, the sensitivity of the metrics across PVT points is shown for all benchmark circuits and test clock speeds under a conventional ATPG pattern set. Results from the TAA pattern set are very similar and are not shown here to avoid redundancy. The figures, show that the DTC exhibited the least changes (least sensitive), while WeSPer is the most sensitive in general.

To know whether the sensitivity of a metric is justified, we correlate the change in metric value to the change in the MSD value, since the MSD value reflects the average size of the test-escape window in a test. As mentioned earlier in section 3.4, an ideal test would have a zero MSD. Thus, metrics should have negative correlation with the MSD in order to justify their sensitivity to PVT variations. In Fig. 3.8, the correlation between the metric change and MSD change across PVT points is shown. In terms of correlation to MSD change, the

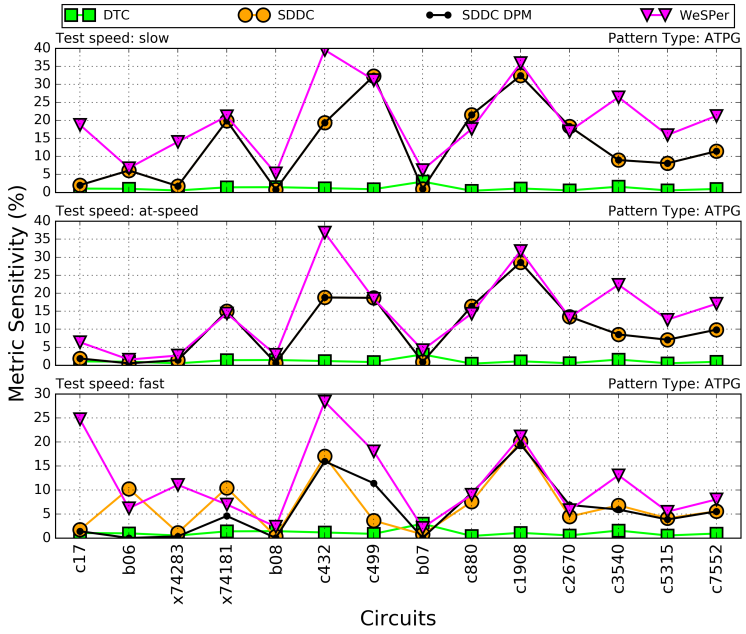

Figure 3.7 The sensitivity of each metric across PVT points plotted for three test clock speeds when using the ATPG pattern set.

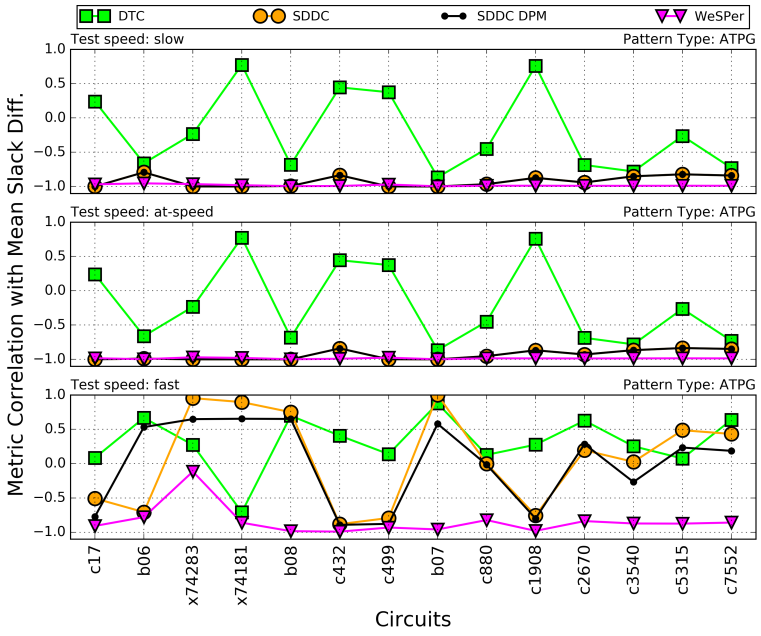

Figure 3.8 Correlations between metric changes and MSD changes across PVT points for three test clock speeds: slow (top), at-speed (middle) and fast (bottom). Note that a negative correlation is expected, thus the closer the result is to -1 the better.

DTC has the worst overall correlation, while WeSPer has the best correlation. This leads to the conclusion that WeSPer is the most sensitive and most correlated to the test-escape window size among the selected metrics.

3.4.5 Predictability

As mentioned in section 3.3, due to the involved extraction of detailed delay information for each fault in the circuit, computing WeSPer over a large set of PVT points can be time consuming and not practical, especially when a circuit design goes through several iterations. Thus, if the metric value can be predicted with acceptable accuracy from a smaller set of PVT points, rather than fully computed from the whole data set, a great amount of design time would be saved. To assess the predictability of WeSPer across the full set of PVT points, WeSPer is plotted under process, voltage and temperature variations separately. Fig. 3.9 plots WeSPer mean and standard deviation values against one dimension of PVT change at a time (process, temperature or voltage). For better visibility of the results, only a set of selected benchmark circuits are plotted (c432, c3540, c5315, c7552). The colored bars represents the mean value of WeSPer, whereas the smaller black bar show the standard deviation value. The plots show only the results for the conventional ATPG pattern set and slow test clock, however the following conclusion holds true for all cases.

In the case of process point variation, the mean WeSPer value has a slight linear increase of no more than 10% as the process point goes from slow to fast (Fig. 3.9). Also, the mean value of WeSPer is almost constant (less than 5% variation) with the change of temperature. However, under voltage variations, the mean value of WeSPer has up to 23% change. Thus, we will focus on predicting the change in the mean value of WeSPer with the change of the voltage point. It can be seen in Fig. 3.9 that, for all circuits, the change of the mean value of WeSPer with voltage follows the same exponential trend. In this case, the voltage curves can be predicted with 3 PVT points samples instead of 54. This saves drastically the computation time for metrics across a large number of PVT points. Fig. 3.9 shows the prediction of the curve using three PVT points (typical point and $25°C$ and voltages $0.6V$ $0.8V$ and $0.95V$) in a dashed line. The curve fit was obtained using the following equation: $a - be^{-cV}$. The average percentage error between the fit and the mean data is less than 1.5% for all benchmark circuits. This means that we can predict WeSPer mean values with an acceptable accuracy for any circuit across all PVT variations with only 3 PVT points. This result holds true for all test speeds and pattern set types.

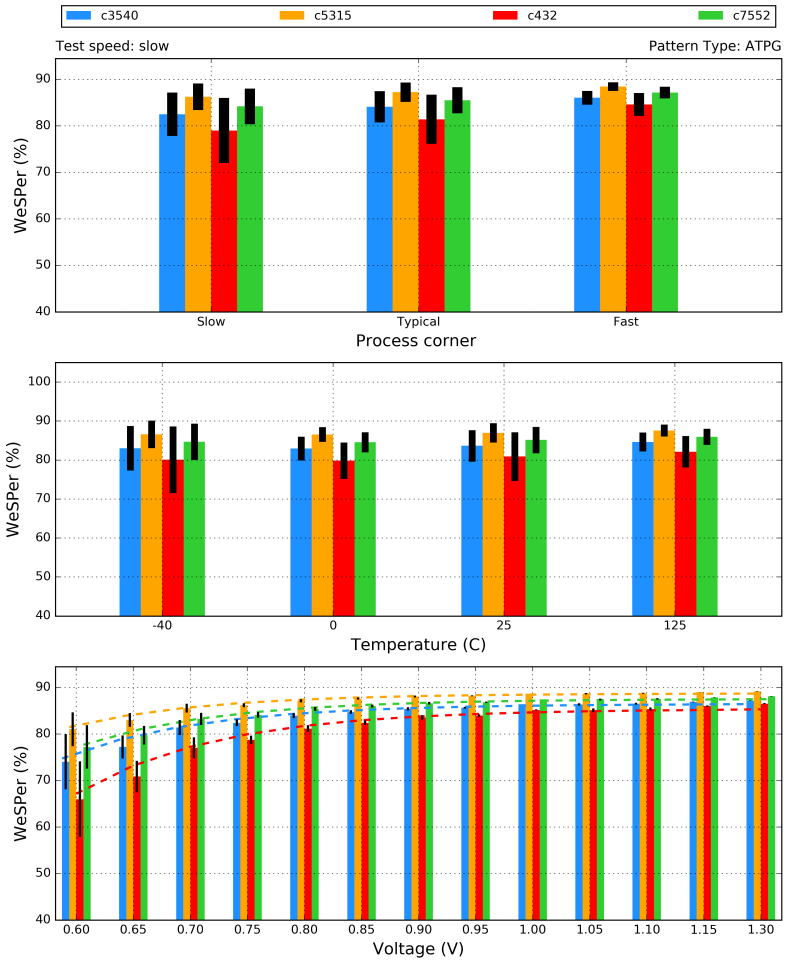

Figure 3.9 WeSPer values with the change of process point, temperature and voltage for a selected set of circuits. The bars in the figure are ordered as in the legend and represent the mean value. The standard deviation value is represented by the smaller black bars. The dashed line shows that a very good curve fit can be obtained with only 3 PVT points using the equation $a - b\,exp(-c\,V)$.

On the practical side, by looking at the standard deviation (black bar) in the voltage subplot in Fig. 3.9, it can be seen that higher voltage points have negligible variation in the WeSPer value across process and temperature changes. Thus, applying the SDD tests on circuits should be done with higher supply voltage since the test quality is consistent over all process points and temperatures.

3.4.6 Worst-Case PVT Point

Another conclusion that can be reached from observing the metric change with PVT variations in Fig. 3.4 and 3.9 is that the worst-case value for WeSPer is observed when testing at

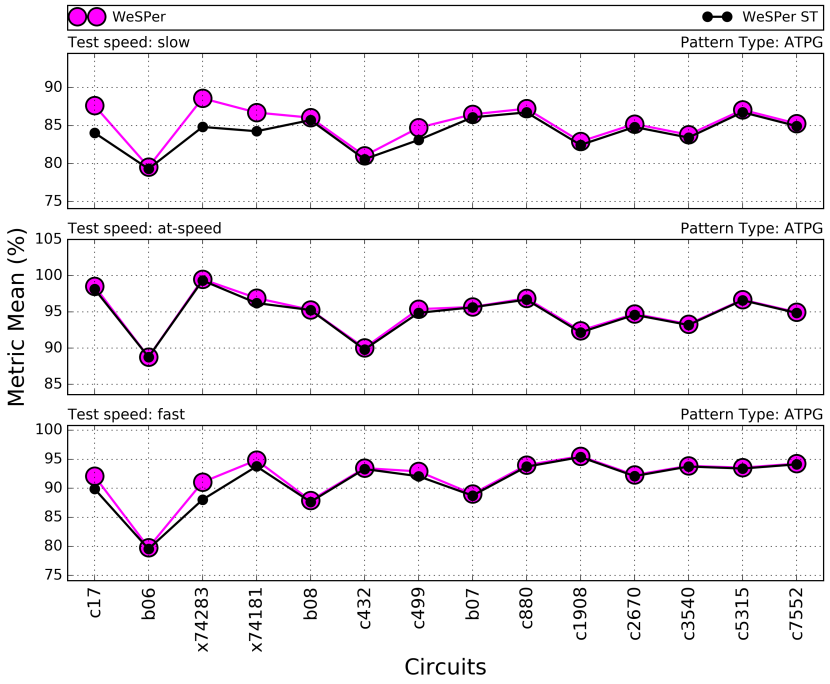

Figure 3.10 Comparison between non-statistical WeSPer and statistical WeSPer (WeSPer_{ST}) under an ATPG patter and three different clock speed.

the slowest process point, the lowest voltage and the lowest temperature. This means that if a conservative estimation of the SDD test quality is to be made, it has to be done at that PVT point. In other words, if only one value needs to be reported as the SDD test quality, the worst case test should be reported since testing at any other PVT point will always yield a better result. Notice that this result could appear counterintuitive if one thinks that slower PVT points would imply longer path delays and thus better SDD test. The results shows that this is not the case. The test-escape window size is actually larger at those PVT points and thus the SDD test quality is worse. Also, note that the fact that the slowest PVT point happens at the lowest temperature is also counterintuitive. This is due to an effect called the inverted temperature dependence [78], where at lower voltages, the lower the temperatures the slower the circuit. This is opposite to what is known at regular voltages.

3.4.7 Statistical WeSPer

In section 3.2.2 we introduced the possibility of making WeSPer a statistical metric by defining a statistical CL multiplier (CL_{ST}). Note that throughout this work, WeSPer results that use the CL_{ST} in the calculation are labeled as WeSPer_{ST}. Fig. 3.10 plots the mean value of statistical WeSPer (WeSPer_{ST}) and the regular WeSPer over all PVT points. It can

be seen that the effect of CL_{ST} is minimal on the value of WeSPer since there is less than 5% change in value. In the case of at-speed testing, there is hardly any difference in the result, whereas for other test clock speeds, the difference is more apparent in smaller circuits. CL_{ST} increases the sensitivity and accuracy of WeSPer if an accurate delay defect distribution is available. However, since the delay defect distribution for the technology that we are using is not available, and since its effect on WeSPer value is minimal, the non-statistical version of WeSPer is considered the preferred option for this work.

3.5 Summary

In this chapter, a new SDD test quality metric, called the Weighted Slack Percentage was presented. WeSPer is a flexible metric that was formulated to represent the quality of the applied SDD test with respect to the newly presented ideal SDD test model. To focus on testing SDD that fail the circuit under normal operating conditions, an overtesting CL multiplier for WeSPer was presented in this chapter. WeSPer is a non-statistical SDD test quality metric that does not depend on the availability of a delay defect distribution. However, if an accurate delay defect distribution is available, the statistical CL multiplier that was developed in this chapter, can be added to allow WeSPer to leverage this information and increase its accuracy. A complete computation flow for WeSPer, and other selected SDD test quality metrics, was presented in this chapter. The importance of PVT variations on the SDD test quality was studied in this chapter and WeSPer was compared with other metrics over 54 PVT simulation points. The results in this chapter have shown that WeSPer is more suitable than other metric in estimating the SDD test quality with the change of test clock speeds. Moreover, the results have shown that WeSPer is the most accurately sensitive metric to delay changes due to PVT point change. The results have also demonstrated that adding a delay defect distribution to WeSPer only changed the test quality value by no more than 5%. Lastly, this chapter also present two methods to estimate WeSPer value over a large set of PVT points. That is, either by looking at the worst case, which is at the slower process point, lowest voltage and lowest temperature, or by curve fitting an exponential equation using three PVT points.

CHAPTER 4 OPTIMIZED SDD TEST BASED ON THE IDEAL TEST MODEL AND WESPER

Thus far the targeted self-timed processor for which a high quality delay test is needed has been presented. The main idea is to take advantage of the CDL to create a high quality delay test. In order to achieve this goal, the ideal SDD test model and formulated WeSPer (an SDD quality metric) were proposed. In this chapter, a new method for generating an optimized faster-than-at-speed test based on WeSPer and the ideal SDD test model is presented. The goal of the proposed optimization algorithm is to improve the quality of a classical TDF delay test while reducing or completely avoiding overtesting the CUT. The proposed technique works on a predefined TDF test patterns and a set of predefined test clock frequencies. It generates the most optimized faster-than-at-speed test by pairing test patterns with test clocks and applying the appropriate masking. In this work, two sets of tests will be generated. One test will completely avoid overtesting and the other will allow overtesting only if it scores a higher WeSPer value. The former test completely avoids catching reliability defects and focuses only of effective SDDs (this is equivalent to a high quality at-speed test), while the latter includes reliability defect if they are close to failing the circuit (their size is close to the size of the SEDD). The proposed optimization technique is first presented, then validated on a selected set of benchmark circuits implemented in a 28nm FD-SOI technology. A discussion about some of the main optimization parameters will follow based on those results.

4.1 Test Optimization

The proposed optimization technique uses a regular TDF based test pattern set and a set of existing test clock speeds to generate an optimized faster-than-at-speed test based on WeSPer. For each fault, the optimizer will pair a single test pattern with a test clock speed and an endpoint mask vector to maximize WeSPer. To reduce the time of applying the optimized test, faults with similar test conditions (test clock speed, pattern and masking) must be grouped together. In order to reduce the final groups count, the optimization process will consider multiple possible test options for each fault.

The optimization flow goes into several steps until the final optimized pattern set is found (Fig. 4.1). The flow is divided into two category of stages: setup stages and optimization stages. The setup stages collect the delay information needed for the optimization, whereas in

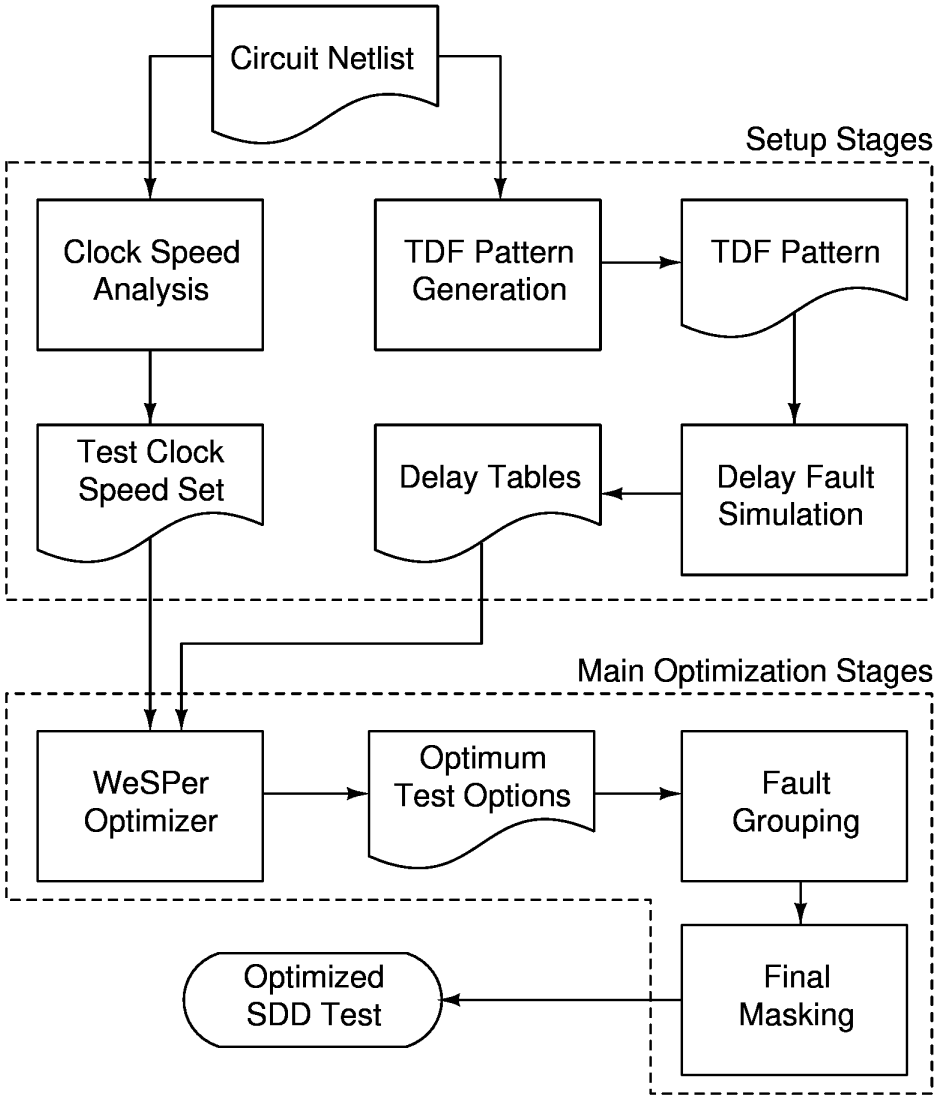

Figure 4.1 SDD test optimization process flow diagram

the main optimization stages, computations are run on the collected information to generate the optimized SDD test.

In the setup stages, the first step is to generate a TDF test pattern set for the CUT and then extract delay information through fault simulation. At the moment, this flow is only compatible for a single PVT point, and thus the delay tables extracted are from a single PVT point. As in the framework presented in chapter 3, an industrial test tool (Tessent from Mentor [74]) is used to extract the longest true path delay for each fault and the activated path delay. The information is organized into a two-dimensional table for the longest true path delay and a three-dimensional table for the activated path delays. The longest true

path delay table lists delays of the longest path through each fault to every endpoint in the circuit, whereas the activated path delay table lists the delay of paths through each fault to every endpoint for each pattern (as shown in Fig. 3.3).

Note that due to limitations in the used tool, the path delay through each fault is reported without any topological information about that path. Thus the proposed optimization only uses the delay information extracted from the available industrial tool and no in-house tool is developed to do any kind of path search or path analysis. In parallel with the TDF pattern generation, a timing analysis is done on the CUT to get the available test clock speeds. Once the delay information of the CUT and the test clock speeds are extracted, the information is passed to the main optimization stages. The main optimization stages are all implemented in Python.

4.1.1 Single Fault Optimization

The delay information for each fault in the CUT, as well as the test clock speeds, are passed to the WeSPer optimizer. The goal of this stage is to find the optimum test conditions (test pattern, test clock speed, endpoint masking vector) for each fault. However, if the test is built with a single test option for each fault, it will be difficult to group faults together due to conflicting test conditions. Thus, in order to reduce the pattern count later, this stage generates a given number of test options for each fault. This number is referred to in this work as the *depth* of the test. As the test options are generated, they are ordered based on the single fault WeSPer value, from high to low. This ordering helps later to reduce the final pattern count while keeping the delay test quality (WeSPer value) high.

During the selection of test options, the test options that result in overtesting can be either considered or completely avoided. In the first scenario, an overtested option is selected over an undertested one only if it has a better single fault WeSPer score. This normally implies that the size of the overtested fault is close to the SEDD, and thus, it is close to failing to the circuit. The second scenario is when overtested options are completely avoided. This kind of test will only fail if an effective SDD is caught. In this work both type of tests will be generated and the results will be compared in section 4.2.

To explain how the WeSPer optimizer generates test options, we will first define some math notations to describe the different elements of test. Let T_{clocks} be the set of valid test clock speeds (periods), Ψ be the pattern set used to test the CUT, E be the set of endpoints of

1. **for each** $\phi_i \in \Phi$ **do**
2. $O_i \leftarrow \varnothing$ // new empty set of valid test options
3. **while** $|O_i| \leq$ desired test depth **do**
4. $M_j \leftarrow \varnothing$ // new empty set of max W_i test options
5. **for each** $\tau \in T_{clocks}$ **do**
6. $Q_i \leftarrow \{(\psi, e) : PD^i_A(\psi, e) \in \mathbb{R}_{>0}\}$ // valid data points
7. $Q_p \leftarrow \{(\psi, e) : (\tau, \psi, e, X) \in O_i\}$ // previous options
8. $Q_n \leftarrow \{(\psi, e) : \tau - PD^i_A \leq 0\}$ // ngative slack points
9. $Q_\tau \quad Q_i - Q_p - Q_n$
10. **if** overtesting not allowed **then** // remove overtested points
11. $Q_o \leftarrow \{(\psi, e) : \tau - PD^i_A(\psi, e) < SEDD_i\}$
12. $Q_\tau \quad Q_\tau - Q_o$
13. **end if**
14. **if** $Q_\tau = \varnothing$ **then** // no points meet conditions
15. **continue** // Next test clock
16. **end if**
// find points with max W_i for current τ
17. $\hat{m}_\tau \quad \max_{(\psi,e)\in Q_\tau} (W_i(\tau, \psi, e))$
18. $M_\tau \leftarrow \{(\tau, \psi, e) : W_i(\tau, \psi, e) = \hat{m}_\tau\}$
19. $M_j \quad M_j \cup M_\tau$
20. **end for**
21. **if** $M_j = \varnothing$ **then** // No test option found for any τ
22. **break** // Stop generating test options
23. **end if**
// find the best test clock/data point
24. $\hat{m}_a \quad \max_{(\tau,\psi,e)\in M_j} (W_i(\tau, \psi, e))$
25. $(\hat{\tau}, \hat{\psi}, \hat{e}) \leftarrow (\tau, \psi, e) \in M_j$ **and** $W_i(\tau, \psi, e) = \hat{m}_a$
// generate masking
26. $X_n = \{e_n : \tau - PD^{all}_A(\hat{\psi}, e_n) < 0\}$ // negative slack endpoints
27. $X_o = \{e_o : PD^i_A(\hat{\psi}, e_o) > PD^i_A(\hat{\psi}, \hat{e})\}$ // endpoints testing smaller faults
28. $\hat{X} \quad X_n \cup X_o$
// add option to option list
29. Add $(\hat{\tau}, \hat{\psi}, \hat{e}, \hat{X})$ to O_i
30. **end while**
31. **end for**

Figure 4.2 Test options generation algorithm

the CUT and Φ be the set of all faults in the CUT. For each fault ($\phi_i \in \Phi$) in the CUT, if the fault is testable, there exist a set of valid test options $O_i = \{o^1_i, o^2_i, ..., o^q_i\}$. Each test option $o^j_i \equiv (\tau, \psi, e, X)^j_i$ is a valid test for fault ϕ_i where this test applies a test clock period

of $\tau \in T_{clocks}$ and a test pattern $\psi \in \Psi$ while observing the result at endpoint $e \in E$ and masking all endpoints in X. Note that X is called a masking vector, $X \subset E$ and $e \notin X$. For each fault ϕ_i, the size of the set of valid test options ($|O_i|$) depends on the fault location and the complexity of the CUT.

Let us also define $PD^i_A(\psi, e)$ as the value of the activated path delay extracted from the delay tables for a given fault ϕ_i, pattern ψ and endpoint e. Similarly, a single fault WeSPer value is defined as $W_i(\tau, \psi, e)$, where τ is the test clock period. This implies that the single fault WeSPer is calculated with the following equation:

$$W_i(\tau, \psi, e) = f_i \, CL_i \tag{4.1a}$$

$$f_i = \frac{T_{sys} - PD_{LT_i}}{\tau - PD^i_A(\psi, e)} \quad \text{for} \quad \tau > PD^i_A(\psi, e) \tag{4.1b}$$

where CL_i, and CL_{OTi} are as defined previously in Eq (3.2) and Eq (3.3), respectively. Also, let $W_i((\tau, \psi, e)^j)$ or simply W^j_i denote the single fault WeSPer value for test option o^j_i.

To produce the set of optimum test options, the WeSPer optimizer uses the algorithm described in Fig. 4.2. This algorithm generates a number of test options for each fault in the CUT depending on the desired test depth. For each test option, a search for the test clock, pattern and endpoint that results in the maximum WeSPer is executed. The search looks in the delay tables for data points that are valid, while having a positive slack and have not been selected previously for the current fault. If overtesting is not allowed, data points that result in a test slack smaller that the SEDD are removed from the considered data point set. The algorithm looks for the maximum $W_i(\tau, \psi, e)$ over all test clocks and then generates the required mask.

There are two sets of endpoints that need to be masked in every test option. First for the chosen pattern and for all faults, endpoints with a negative slack are masked. Next for the targeted fault and selected pattern, endpoints with a longer activated path than those reaching the chosen endpoint are masked. This avoids testing a fault smaller than the selected one. Finally, the selected test clock, pattern, endpoint and masking vector are added (as a test option) to the set of optimum test options for the targeted fault.

To facilitate the understanding of this algorithm, Fig. 4.3 visualizes the test option generation and selection process on a single fault example. The figure shows the single fault

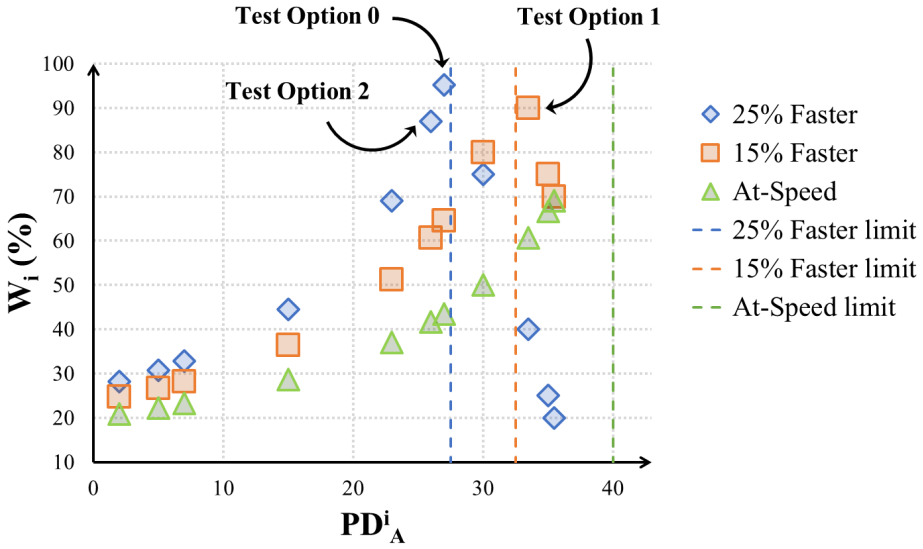

Figure 4.3 Example of the process of generating a set of optimum test options for a given fault and pattern. Options that are under the limit are undertesting the fault whereas ones over the limit are overtesting it.

WeSPer value for a case of a given fault ϕ_i, a given pattern, 11 different endpoints, and a set of 3 test clocks: at-speed, 15% faster-than-at-speed and 25% faster-than-at-speed. The single fault WeSPer value is plotted against the activated path delay for the 3 test clocks. For each of the test clocks, an associated limit is drawn to show when a test option is overtesting the fault. In the selection process, the optimizer lists all the valid test options for each test clock and then selects the one that maximizes W_i. If overtesting is allowed, and the test depth is set to be 3, the three shown options would be selected and listed in the test option file in that order. For each of the selected test options, the masking vector X will include any endpoint that sees a longer activated path delay that the selected one. For test option 2, for example, the masking vector includes all points over the "25% Faster limit" along with the endpoint labeled as "Test option 0". Lastly, note that, if overtesting is not allowed, the test option labeled "Test Option 1" is excluded, and another test option is selected instead.

After the generation of the optimum test options for all faults in the CUT, the maximum overall WeSPer is computed as:

$$WeSPer_{max} = \frac{1}{N}\sum_{i=1}^{N} \max_{1\leq j\leq |O_i|}(W_i^j) \times 100\% \tag{4.2}$$

Note that in this chapter, we are only considering the overtesting confidence level multiplier (CL_{OT}). As a result, $\max(W_i^j)$ can be found for fault ϕ_i by finding the value of the activated path delay ($PD_A^i(\psi, e)$) that makes the tested delay defect closest to the SEDD size. This implies that for a given τ, in the undertested region ($f_i \leq 1$), $W_i(\tau, \psi, e)$ is maximized when $PD_A^i(\psi, e)$ is maximized. Whereas in the overtested region ($f_i > 1$), $W_i(\tau, \psi, e)$ is maximized when $PD_A^i(\psi, e)$ is minimum.

4.1.2 Fault Grouping

The goal of this stage is to create groups of faults that are tested under the same test conditions. The final number of fault groups is equivalent to the final pattern count. However, to avoid confusion with the pattern count of the initial TDF pattern set, in this work, the final pattern count will be referred to as the *groups count*.

To reduce the time of applying the final optimized SDD test, the groups count has to be reduced. This was the motivation to produce multiple test options for each fault in the previous stage. However, since not all test options have maximum single fault WeSPer, the goal of the proposed fault grouping algorithm is to reduce the groups count without penalizing the overall test quality too much. The overall test quality ($WeSPer_{opt}$) is calculated as:

$$WeSPer_{opt} = \frac{1}{N} \sum_{i=1}^{N} (W_i^{j*}) \times 100\% \tag{4.3}$$

where N is the total fault number in the CUT and W_i^{j*} is the single fault WeSPer value of the test option selected by the fault grouping algorithm of fault ϕ_i. Note that W_i^{j*} for untestable faults is zero.

In the previous stage, the WeSPer optimizer outputs the *optimum test options file* (shown in Fig. 4.4). This file lists a number of top test options for a given fault ordered by the achieved single fault WeSPer value. With each of these options, additional information is saved to facilitate computing $WeSPer_{opt}$, grouping faults, and debugging the optimizer. One of the saved additional information that will be used in this stage is labeled as the *Drop*. It signifies the difference between the maximum single fault WeSPer value for a given fault ϕ_i and the single fault WeSPer value of a test option o_i^j. Since test options are ordered in the file based on their single fault WeSPer value, the maximum single fault WeSPer value for fault ϕ_i is W_i^0, and so, the drop is simply computed as:

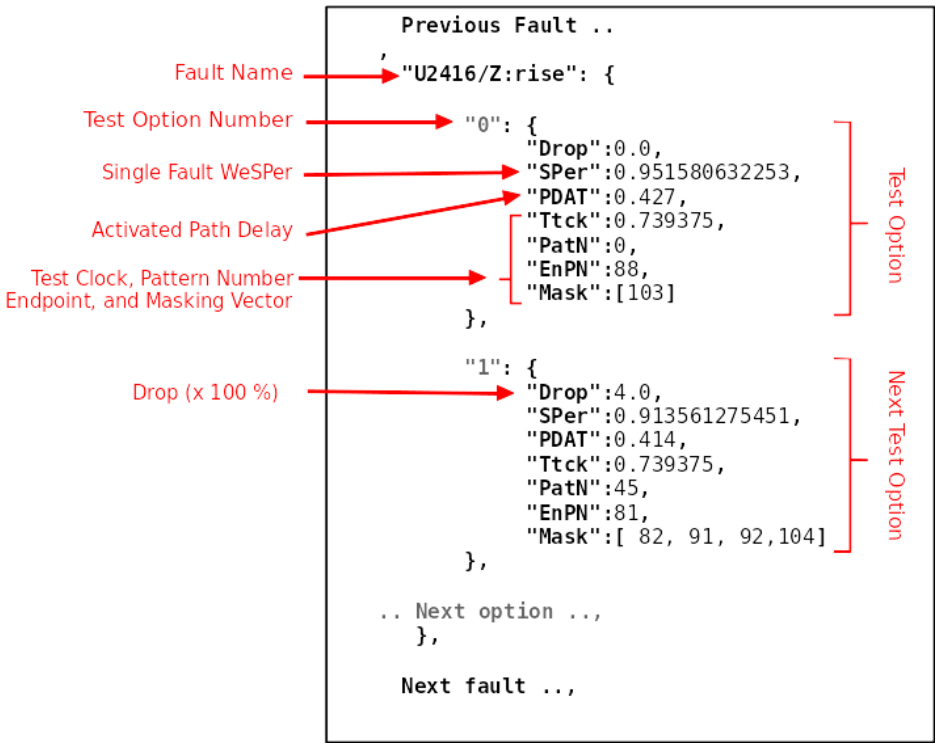

Figure 4.4 Optimum test options file structure written in JavaScript Object Notation (JSON) format.

$$Drop_i^j = W_i^0 - W_i^j \tag{4.4}$$

The simplified algorithm for grouping faults is shown in Fig 4.5. The process of grouping starts by reading the set of testable faults and their test options, then creating a list of the testable faults. For each fault, the algorithm loops over the available test options for the given fault and processes the options in order. As mentioned earlier, the options are ordered from the option with the best quality (single fault WeSPer) and then lower. Hence, the algorithm first tries to group faults with their best option first, and if it is not successful, it moves to the next test option until there is no more test options with an acceptable drop. If the fault fails to group with any existing fault group, the algorithm creates a separate group for this fault with its best test option (o_i^0).

A test option in the option list of a fault is only considered for grouping if it meets a certain quality criteria. The threshold that limits the drop in quality is named the *quality*

1. Φ_G {All testable faults}
2. **while** $\Phi_G \neq \varnothing$ **do**
3. Select $\phi_i \in \Phi_G$
4. **for** $0 < j < |O_i|$ **do**
5. $(\tau, \psi, e, X) \leftarrow o_i^j$
6. **if** $Drop_i^j \leq$ Quality Drop Tolerance **then**
7. **if** There is no group for τ and ψ **then**
8. Create group $G_{\tau,}^1$
9. Add ϕ_i to group $G_{\tau,}^1$
10. Remove ϕ_i from fault list Φ_G
11. **else** // A group for τ and ψ exists
12. **if** (τ, ψ, e, X) is compatible with any existing group **then**
13. Add ϕ_i to the compatible group
14. Remove ϕ_i from fault list Φ_G
15. **else** // o_i^j cannot be grouped with existing groups
16. **if** $j = |O_i|$ **then** // last option failed to group
17. $(\tau, \psi, e, X) \leftarrow o_i^0$ // select best option
18. Create another group $G_{\tau,\psi}^y$
19. Add ϕ_i to group $G_{\tau,\psi}^y$
20. Remove ϕ_i from fault list Φ_G
21. **end if**
22. **end if**
23. **end if**
24. **end if**
25. **end for**
26. **end while**

Figure 4.5 Fault grouping algorithm

drop tolerance. As long as $Drop_i^j$ is lower than or equal to the specified quality drop tolerance, option o_i^j is considered as a valid option for grouping. Each fault group ($G_{\tau,\psi}^y$) is defined by one test clock ($\tau \in T_{clocks}$), one pattern ($\psi \in \Psi$), a set of targeted endpoints ($E_{\tau,\psi}^y$) and a masking vector ($X_{\tau,\psi}^y$). More precisely, $E_{\tau,\psi}^y$ is the set of endpoints targeted by the test options of the faults in the group $G_{\tau,\psi}^y$, whereas $X_{\tau,\psi}^y$ is the union of all the masking vectors of the test options selected to test faults in the same group $G_{\tau,\psi}^y$. Where y is an integer iterator that labels different groups with the same test clock and pattern. When grouping a fault (ϕ_i) into an existing group with the same τ and ψ, the endpoint and masking vector of the selected option $(\tau, \psi, e, X)_i^j$ of fault ϕ_i should be compatible with the group's targeted endpoints ($E_{\tau,\psi}^y$) and masking vector ($X_{\tau,\psi}^y$). If we define e_i^j and X_i^j as the endpoint and masking vector in a selected test option $(\tau, \psi, e, X)_i^j$, then the compatibility condition between that

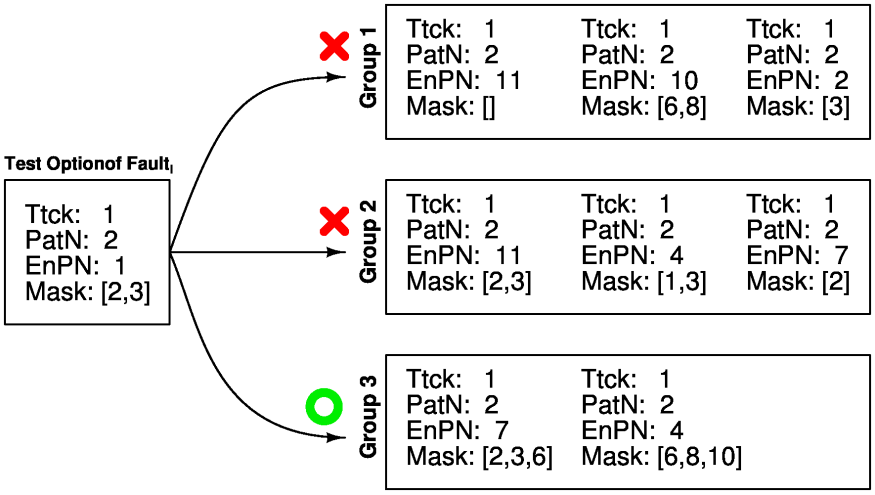

Figure 4.6 Example on fault grouping compatibility.

test option for fault ϕ_i and a fault group $G^y_{\tau,\psi}$ can be written as $\phi_i \in G^y_{\tau,\psi} \Rightarrow e^j_i \notin X^y_{\tau,\psi}$ and $X^j_i \cap E^y_{\tau,\psi} = \varnothing$.

Fig 4.6 shows a simple example on fault grouping and the compatibility condition between test options. First, notice that all test options have to be grouped with other test options that use the same pattern and test clock. In this example, the test option that needs to be grouped conflicts with the first two groups but is compatible with the last group. In first group, the last test option requires its fault to be observed at endpoint 2, however, that endpoint is masked in the introduced test option. Similarly, the test option that needs to be grouped observes the fault effect at endpoint 1, while the group 2 masks that endpoint. Group three satisfies the compatibility condition, and hence, the test option of the new fault can be added to this group.

Once all faults are put into groups, the faults list becomes empty and the process stops. The fault groups are then passed to the final masking stage. This stage applies additional masking on the output pattern based on the group information. The final groups which pairs patterns with clock speed and masking vector compose the final optimized SDD test file (Fig. 4.7).

4.1.3 The Masking Strategy

In this work, three levels of masking are applied to avoid having false failures. The first level of masking is done in the single fault optimization stage where negative endpoints are

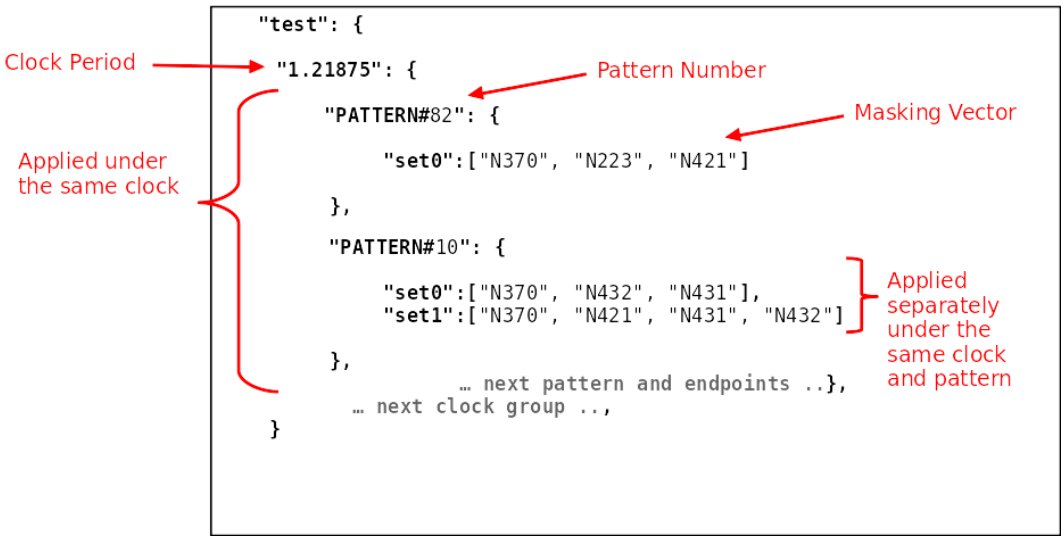

Figure 4.7 Optimized SDD test file structure written in JSON format.

masked. Although this is done for each test option generated, it only depends on the selected test clock speed and pattern. This means that this type of masking does not affect fault grouping since each group uses the same test clock and pattern.

The second level of masking is also done in the single fault optimization stage, where any endpoint that tests a fault of a size bigger than the selected one is masked. This masks overtested endpoints for a given fault. This kind of masking forces us to check the compatibility between masking vectors and targeted endpoints for the faults in a same group. If a conflict exist, another group with same test clock and pattern is created to resolve this conflict.

The last masking level is done during the writing of the final test file. For each fault group, all the endpoints are masked except for the ones in the set of targeted endpoints of the group. This is an extra step to make sure that only the selected endpoints capture the test results.

4.2 SDD Test Optimization Results

This section presents results extracted from selected benchmark circuits that show the properties of the proposed SDD test optimization technique and the relationship between different parameters, such as the test depth, group count and $WeSPer_{opt}$. All of the circuits in this work are implemented using STMicroelectronics FD-SOI 28nm technology.

Table 4.1 Benchmark circuits characteristics

Circuit	# of faults	# of endpoints	# of patterns ATPG	# of patterns TAA	TDFC (%) ATPG or TAA	TAA Average PD_A increase (%)
c432	812	7	85	85	98.77	23.23
c499	934	32	100	124	99.14	2.90
c880	1556	26	85	99	100	14.77
c1908	2048	25	161	190	99.66	10.27
c2670	3268	140	125	175	97.71	7.76
c3540	4862	22	231	250	97.90	3.23
c5315	8028	123	144	179	99.41	10.98
c7552	10298	108	191	228	98.72	3.74
c6288	12322	32	87	89	99.95	1.43

4.2.1 Benchmark circuits results

The SDD test optimization flow described in Fig. 4.1 is applied on a set of benchmark circuits from the ISCAS89. The characteristic of the synthesized design of each of these circuits is summarized in table 4.1. The circuits are ordered in the table by the number of faults from low to high. For each circuit, two types of pattern sets are generated: a non-timing aware TDF pattern set, and a timing-aware one. These pattern sets will be referred to in this work as an ATPG pattern set and a TAA pattern set, respectively. The TDFC is listed for both pattern sets. As mentioned earlier, the TDFC represents the percentage of delay testable faults in the circuit and is not suitable to assess SDD test quality. Still, the TDFC is useful to determine the upper limit that an SDD test can achieve for a give pattern set. Additionally, the last column of table 4.1 lists the average increase in PD_A (activated path delay). This helps to show the improvement in sensitizing longer paths due to using a timing-aware pattern set.

Table 4.2 shows the results of the WeSPer optimization stage for each benchmark circuit. In these results, the T_{sys} is set to be 25% larger than the critical path delay of the circuit, whereas for the test clock periods, two scenarios are explored. The first uses 4 test clocks with periods being $1\times$, $0.9\times$, $0.8\times$ and $0.7 \times T_{sys}$, where as the second scenario uses 7 test clock periods being $1\times$, $0.95\times$, $0.9\times$, $0.85\times$, $0.8\times$, $0.75\times$and $0.7\times T_{sys}$. Notice that in each of these test clock sets, there is one at-speed test clock. This is essential to reduce or completely avoid overtesting longer paths. Also, notice that the choice of the 7 test clock periods is on the same range of the 4 test clock periods but with more selected periods in between. This helps us determine the usefulness of adding more test clocks frequencies in the process of optimization.

Table 4.2 Optimized SDD test quality comparison

	4 Test Clocks → $(1, 0.9, 0.8, 0.7) \times T_{sys}$							
	ATPG				**TAA**			
	At-speed test	**$WeSPer_{max}$ (%)**			**At-speed test**	**$WeSPer_{max}$ (%)**		
Circuit	**WeSPer(%)**	**No Over.**	**With Over.**	**TOPer(%)**	**WeSPer(%)**	**No Over.**	**With Over.**	**TOPer(%)**
c432	60.53	83.60	85.06	0.92	68.81	90.82	92.34	1.18
c499	79.25	87.62	89.73	1.72	84.80	91.26	92.92	1.15
c880	84.43	96.69	97.97	0.59	88.22	97.99	98.92	0.47
c1908	65.92	91.63	92.93	0.60	70.63	94.38	95.59	0.54
c2670	84.05	93.93	94.92	0.71	88.39	95.06	95.74	0.53
c3540	74.75	94.47	95.41	0.52	76.54	95.30	96.13	0.47
c5315	86.14	96.58	97.54	0.56	88.50	97.62	98.27	0.37
c7552	80.79	95.48	96.48	0.66	83.29	96.13	97.00	0.57
c6288	52.92	85.03	85.86	0.34	54.07	86.89	87.78	0.46
	7 Test Clocks → $(1, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7) \times T_{sys}$							
	ATPG				**TAA**			
	At-speed test	**$WeSPer_{max}$ (%)**			**At-speed test**	**$WeSPer_{max}$ (%)**		
Circuit	**WeSPer(%)**	**No Over.**	**With Over.**	**TOPer(%)**	**WeSPer(%)**	**No Over.**	**With Over.**	**TOPer(%)**
c432	60.53	85.68	86.66	0.52	68.81	93.10	93.76	0.55
c499	79.25	91.33	92.32	0.66	84.80	93.92	94.63	0.55
c880	84.43	97.75	98.50	0.38	88.22	98.85	99.37	0.26
c1908	65.92	92.70	93.43	0.37	70.63	95.38	96.09	0.33
c2670	84.05	95.23	95.78	0.36	88.39	96.10	96.61	0.30
c3540	74.75	95.33	95.98	0.32	76.54	96.14	96.70	0.27
c5315	86.14	97.69	98.16	0.25	88.50	98.39	98.74	0.17
c7552	80.79	96.76	97.34	0.31	83.29	97.23	97.75	0.28
c6288	52.92	85.69	86.18	0.20	54.07	87.64	88.17	0.30

For the chosen set of benchmark circuits, the results in table 4.2 show that the test quality (namely the WeSPer value) can improve by up to 34.1% with respect to a classical at-speed test by using the proposed optimization technique (a relative increase of 63.07%). This is the case for the c6288, optimized with 7 test clocks and using timing-aware pattern set. In terms of the effect of avoiding overtesting, the results show that completely avoiding overtesting during the optimization process reduces the overall $WeSPer_{max}$ slightly ($< 2.11\%$). In addition, it can be seen that when overtesting is allowed, the value of TOPer is very small. This indicates that only overtested points with sizes close to the SEDD are considered during optimization. Note that the TOPer value for the case of no overtesting (column 3 and 7) is always zero. The results also show that there is a slight improvement in the optimized SDD test quality when using the 7 test clocks scenario over the 4 test clock one, where $WeSPer_{max}$ increases and TOPer decreases. However, as will be shown later, increasing the number of test clocks could lead to an increase in the groups count, and of course, an increase in the size of the clock generation logic (e.g. CDL). This is an undesirable side-effect for such a slight increase in test quality.

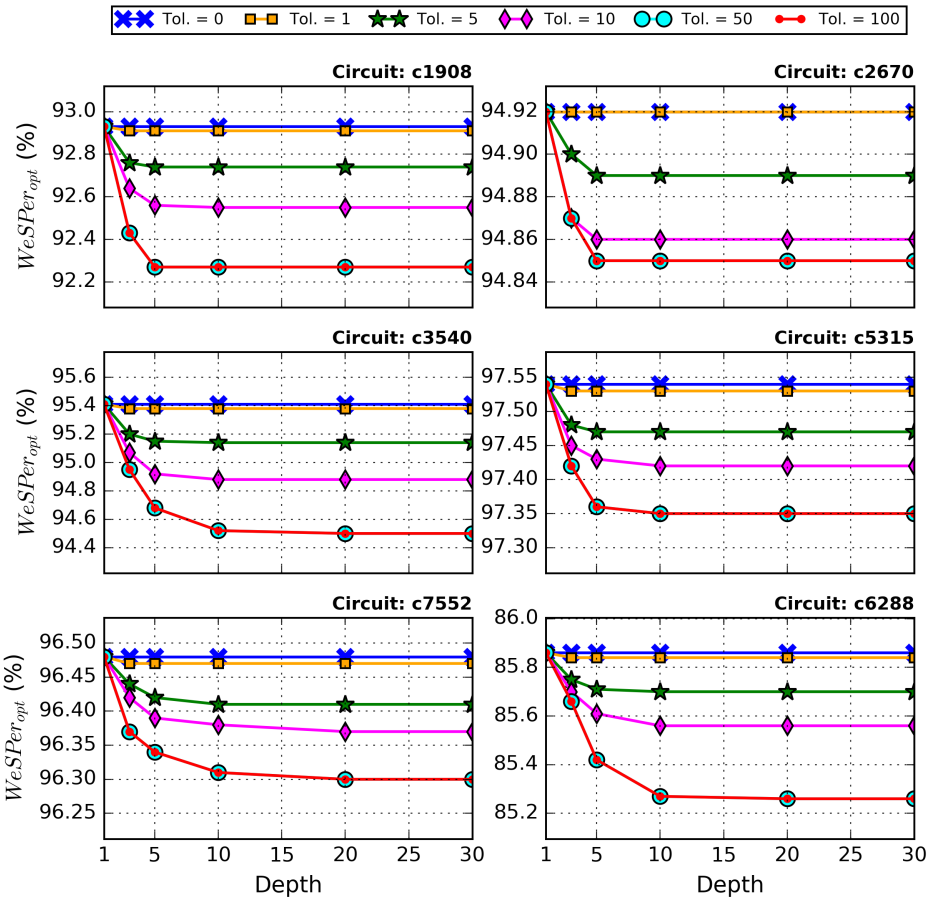

Figure 4.8 Fault grouping analysis on benchmark circuits. $WeSPer_{opt}$ versus the depth of the test options file and quality drop tolerance. This is for the case of 4 test clocks, with non timing-aware pattern set while allowing overtesting. The results of the alternative cases is similar.

As can be seen from table 4.2, a TAA pattern set improves the WeSPer value of the delay test in the case of at-speed testing. However, for the proposed SDD optimization technique, using the TAA pattern set does not always produce a significant increase in $WeSPer_{max}$ value over the ATPG one. The optimized test quality is more dependent on the matching between the available test clock periods and the path delay activated by the pattern set with respect to the SEDD size. It is also interesting to notice that, the optimized SDD test produced by this work with the ATPG pattern set can improve the test quality by up to 32.11% compared to the at-speed test that uses the TAA pattern set (a relative increase of 59.39%).

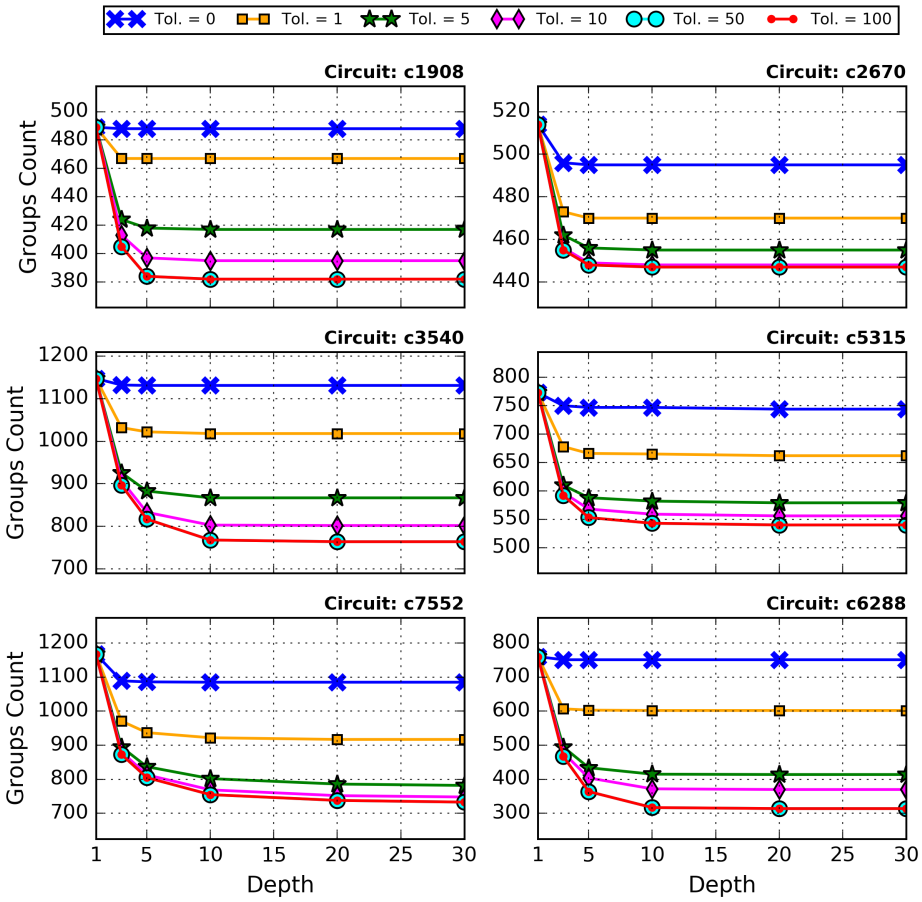

Figure 4.9 Fault grouping analysis on benchmark circuits. Groups count versus the depth of the test option file and quality drop tolerance. This is for the case of 4 test clocks, with non timing-aware pattern set while allowing overtesting. The results of the alternative cases is similar.

The goal of the fault grouping process in the proposed optimization technique is to reduce the final groups count without degrading the final test quality. The proposed grouping method depends on two factors: the depth of the generated test option file, and the quality drop tolerance. As a reminder, the depth relates to the maximum number of optimum test options generated for each fault, while the quality drop tolerance puts a limit on the single fault WeSPer degradation during the test option selecting and grouping process.

Fig. 4.8 and 4.9 show the effect of these two factors on the final test quality ($WeSPer_{opt}$) and the final groups count. Note that these figures are plotted for the case of 4 test clocks, with the ATPG pattern set while allowing overtesting, and that the results of the alternative cases show the same behavior. It can be seen in these figures that both $WeSPer_{opt}$ and the

groups count decrease as the depth and tolerance increase. The general observation here is that the degradation in $WeSPer_{opt}$ is always very small, whereas the reduction in the groups count can be significant. For example, on the c3540, when the depth is set to 30 test options and the tolerance is 100%, $WeSPer_{opt}$ loses less than 1% of quality whereas the groups count drops from 1147 to 764 groups (33% less).

Another general observation can be made here about the relationship of the results to the depth of the test option file. On average, the results saturate around 10 test options, and they are generally constant after 30 test options. This means that generating more than 30 test options per fault for a given circuit does not produce any benefit. Also note that, for the case when the tolerance is 0%, the test quality is constant ($WeSPer_{opt} = WeSPer_{max}$) whereas the groups count can still be reduced if more than one test option is generated. This is because sometimes multiple test options for the same fault can have the same maximum test quality while having different test conditions. Thus, even if the tolerance is set to zero, generating multiple test options for each fault can reduce the final groups count.

Table 4.3 lists the results of the final optimized SDD test quality and the groups count for the case when the test option file depth is 30 test options, the quality drop tolerance is 100% and overtesting is not allowed. This table also lists the percentage of the test clocks and pattern that were used to build the final fault groups. The results presented in this table show that using more test clocks enhances the SDD test quality ($WeSPer_{opt}$) slightly but for the price of an increased groups count. For example, in the case of the c5315 and a TAA pattern set, the $WeSPer_{opt}$ gains 0.9% while the groups count increases 59.28%.

The utilization percentage results show that all the clocks in all the cases were used in creating the fault groups. This means that none of the selected test clock periods were useless. On the other hand, the pattern sets are not always fully used, but the utilization in general is very high ($> 95\%$). In general, the proposed optimization technique was able to improve the test quality by up to 32.97% compared to an at-speed test using the same pattern set (a relative increase of 60.98%).

Now that the proposed optimization technique is verified, the test will be applied on the AnARM chips. The post-silicon optimized test results, as well as the test setup and at-speed test results are shown and discussed in the following chapter.

Table 4.3 The quality and groups count for the optimized SDD test

	4 Test Clocks → $(1, 0.9, 0.8, 0.7) \times T_{sys}$							
	ATPG				**TAA**			
		Groups	**Utilization (%)**			**Groups**	**Utilization (%)**	
Circuit	$WeSPer_{opt}$	**count**	**Patterns**	T_{clocks}	$WeSPer_{opt}$	**count**	**Patterns**	T_{clocks}
c432	83.13	227	97.65	100	90.48	201	95.29	100
c499	87.62	246	100	100	91.26	266	96.77	100
c880	96.48	267	100	100	97.9	287	100	100
c1908	90.65	402	97.52	100	93.78	488	97.37	100
c2670	93.82	440	100	100	95.02	522	99.43	100
c3540	93.22	759	100	100	94.24	811	100	100
c5315	96.24	536	96.53	100	97.37	663	100	100
c7552	95.21	741	99.48	100	95.95	883	100	100
c6288	84.10	327	98.85	100	85.90	348	100	100
	7 Test Clocks → $(1, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7) \times T_{sys}$							
	ATPG				**TAA**			
		Groups	**Utilization (%)**			**Groups**	**Utilization (%)**	
Circuit	$WeSPer_{opt}$	**count**	**Patterns**	T_{clocks}	$WeSPer_{opt}$	**count**	**Patterns**	T_{clocks}
c432	85.57	244	95.29	100	92.94	235	97.65	100
c499	91.33	316	99.00	100	93.92	338	95.97	100
c880	97.64	357	100.00	100	98.82	373	100	100
c1908	92.06	514	98.14	100	95.09	617	97.89	100
c2670	95.16	665	100.00	100	96.07	729	100	100
c3540	94.66	1119	99.57	100	95.6	1181	100	100
c5315	97.51	895	96.53	100	98.27	1056	100	100
c7552	96.6	1244	99.48	100	97.16	1466	100	100
c6288	85.07	541	98.85	100	87.04	568	100	100

Note: This data is for the case when depth = 30, tolerance = 100% and no overtesting is allowed.

4.3 Summary

In this chapter, a new SDD test optimization method that is based on WeSPer and faster-than-at-speed testing was presented. The main idea of this optimization method is to match patterns from an existing pattern set with a test clock speed from a predefined set of at-speed and faster-than-at-speed test clocks to maximize the WeSPer value (SDD test quality) of a test. The proposed method included a fault grouping and endpoint masking algorithm that helped reducing the final pattern count while keeping the SDD test quality high. The proposed optimized SDD test was validated on a set of benchmark circuits synthesized in the 28nm FD-SOI technology. The results show that the proposed algorithm is able to improve WeSPer value up to 32.97% compared to classical at-speed testing using the same pattern set.

CHAPTER 5 POST-SILICON TEST RESULTS OF THE ANARM

Thus far the structure of the AnARM and the need for developing a high quality delay test for it have been explained. The proposed test plan is to exploit the CDL built into the AnARM self-timed structure to apply a high quality SDD test. An at-speed test methodology has been discussed in section 2.6 where a new LOC strategy that takes advantage of the built-in CDL has been presented (section 2.6.3). Moreover, in the previous chapters, an SDD test quality metric and a method of generating an optimized SDD test that exploits the variable test speeds capabilities offered by the AnARM self-timed structure have been developed.

In this chapter, the at-speed test method, including the AnARM LOC strategy, is verified then applied on 25 AnARM chips using a Teradyne FLEX tester. After verifying the functionality of the new at-speed test strategy, the proposed high quality SDD test is applied on the functional AnARM chips and the results are presented and discussed. This chapter also presents the test setup, scripts to interface with the tester and more details about the target circuit inside the AnARM.

5.1 Pre-Silicon Test Setup

In section 2.6, an LOC based at-speed test strategy that works with the AnARM self-timed structure was presented. This test strategy is completely compatible with conventional test and ATPG tools. By using multiplexers in the data and clock paths, the presented strategy allows for conventional test vector scanning and takes advantage of the built-in delay lines to apply at-speed testing. The presented at-speed test method was validated on two circuits that use the targeted Octasic self-timed structure. The first circuit is a small processor that contains 4 arithmetic logic units (called picoALUs). The proposed test concept and implementation flow are validated on the picoALUs with pattern generation, coverage reports and simulation results. The picoALUs serve as a proof of concept and were not fabricated. The second circuit is a set of four 32-bit multiplier units inside the AnARM. The test is applied on the last stage of these multipliers. We will simply refer to these circuits from here on as the Mul-units. The CDLs in the Mul-units have been specially adjusted to allow for the optimized SDD test, as will be shown later.

The technology that is used for all steps of design, test and fabrication is the STMicroelectronics 28nm FD-SOI CMOS technology. Scan chains insertion and pattern generation

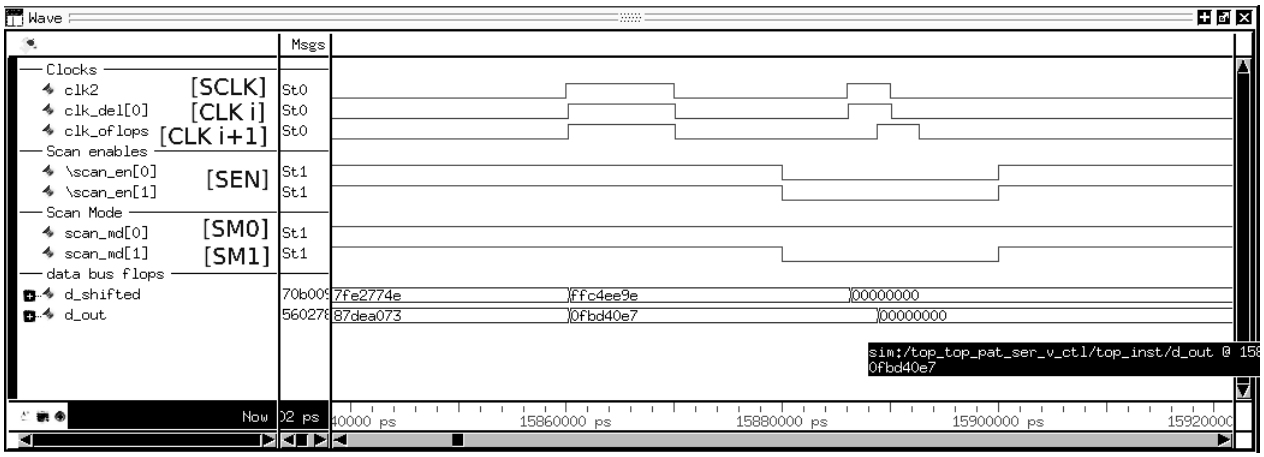

Figure 5.1 Post-synthesis simulation of the proposed LOC on the picoALU showing the transition from the shift mode to the at-speed test mode. The signal names, with reference to Fig. 2.11, are shown next to the original design names.

were done using Mentor Graphics Tessent. Tessent was also used to apply the proposed LOC scheme using the "named capture procedures". The LOC timing is validated with post synthesis gate simulation using a test bench that was automatically generated from Tessent. The simulation of the new LOC strategy on the picoALU shows (in Fig. 5.1) the LOC signals behavior for the transition from the shift mode to the at-speed test mode. Notice that this behavior is identical to the one described in Fig. 2.11. The moment of the launch and capture are indicated in the figure by the rising edges that occur on the clock signals (CLK_i and CLK_{i+1}) when the S_{EN} is low. The details of the input signals of the AnARM chips, along with general information on the applied patterns and time-plates are listed for future reference in appendix B.

5.1.1 At-Speed Test Coverage

The test patterns for the picoALU and the AnARM were generated under the TDF model with a regular (non-timing aware) ATPG engine. The coverage results for the 4 picoALUs and the 4 Mul-units circuits inside the AnARM processors are shown in Table 5.1. The table reports the number of testable faults, the number of faults undetected by the pattern, the number of applied patterns and the TDFC percentage. As reported in the table, an overall of 98.37% TDF coverage was obtained on the picoALU circuits and an overall of 96.07% TDF coverage for the 4 Mul-units in the multiplier units of the AnARM. Notice that the number of patterns in all cases is the same for all Mul-units because the patterns are applied simultaneously. As can be seen in these results, the special LOC method for the AnARM is

Table 5.1 Coverage results

Design Unit Name	**Number of Testable Faults**	**Number of Undetected Faults**	**Number of Patterns**	**TFD Coverage (%)**
PicoALU00	1711	16	213	99.06
PicoALU01	1723	28	213	98.37
PicoALU02	1723	28	213	98.37
PicoALU03	1735	40	213	97.69
Overall: PicoALUs	6892	112	213	98.37
AnARM Mul. Unit00	14118	555	202	96.07
AnARM Mul. Unit00	14118	555	202	96.07
AnARM Mul. Unit00	14118	555	202	96.07
AnARM Mul. Unit00	14118	555	202	96.07
Overall: AnARM Mul. Units	56472	2220	202	96.07

able to achieve high coverage results.

5.1.2 Faster than At-Speed Setup

The CDL in the AnARM self-timed structure opens the possibility to use the same LOC technique to do more than just at-speed testing. The CDL can be designed to accommodate faster-than-at-speed testing, reliability testing and/or speed binning. Moreover, by accommodating shorter delay lines in the CDL, we are able to generate faster-that-at-speed tests that fail the CUT intentionally. By doing so, we are able to validate that the applied AnARM LOC test strategy indeed works when testing fabricated chips, and are able to used it to apply the proposed optimized SDD test method. In terms of the CDL structure, there are different methods to build delay lines, however, in all cases, the added adjustments to the CDL might incur an area overhead on the existing design. Thus, there is a trade-off between the number of delay stages that can be added to the CDL to accommodate additional test features and the area of the design. The CDLs that is integrated in the Mul-units are built from a muxed inverter chain. A simplified schematic of the CDL is shown in Fig. 5.2. Each delay stage in the CDL is series of inverters. The number of inverters in each delay stage depends on the amount of delay needed for each stage. The number of delay stages in the CDL depends on the number of test clock speeds needed. In the Mul-units, we decided to test these circuit with 4 test clock speeds that are controlled externally by two fault-injection signals (FI_0 and FI_1). Depending on these two signals, we are able to apply delay testing with an at-speed clock and 3 speed levels of faster-than-at-speed clocks. The decision to set the 3 fast test clocks was made before developing the optimized SDD test method. The test clock periods were spaced equally inside the path delay distribution of the Mul-units. To do so, a timing report of the delays of the 200,000 longest paths in the circuit was generated

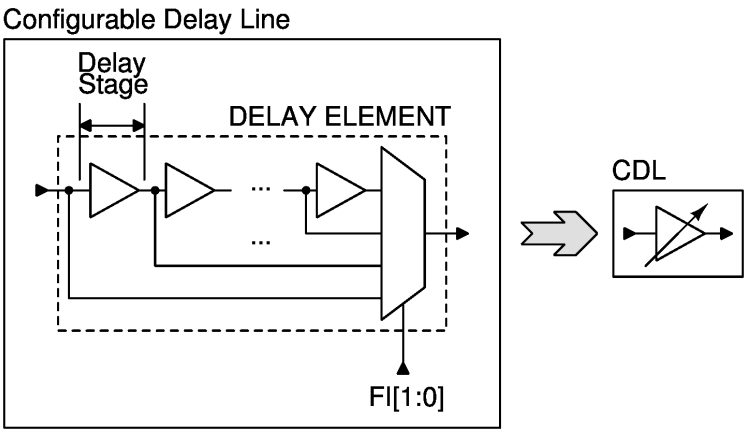

Figure 5.2 Configurable delay line (CDL) structure.

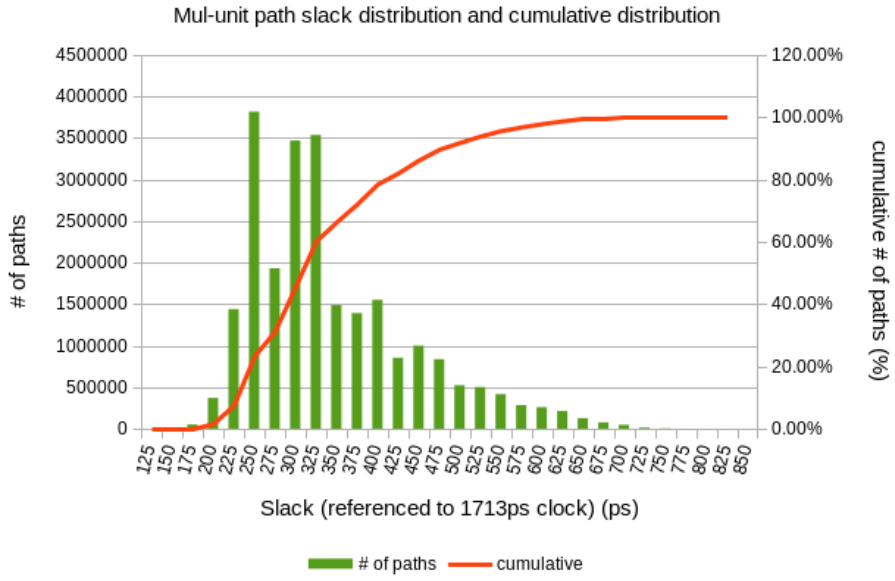

Figure 5.3 The slack timing report showing the path slack distribution of 200,000 paths in the Mul-unit circuit. This report is generated at the expect PVT point (TT/0.9V/25°C).

using a timing analysis tool. The 3 fast clock speeds of the CDL were chosen to be faster than 25%, 50% and 75% of the reported paths when FI_0 and FI_1 are 01, 10 and 11, respectively. Normal at-speed testing is activated by setting FI_0 and FI_1 to 00. The path delay report and the test clock speeds selected are shown in Fig. 5.3. The fast clock speeds are selected with respect to the cumulative number of paths. The post synthesis timing analysis of the Mul-units showed that the CDL delays are set to be 1.713 ns, 1.438 ns, 1.338 ns and 1.313 ns.

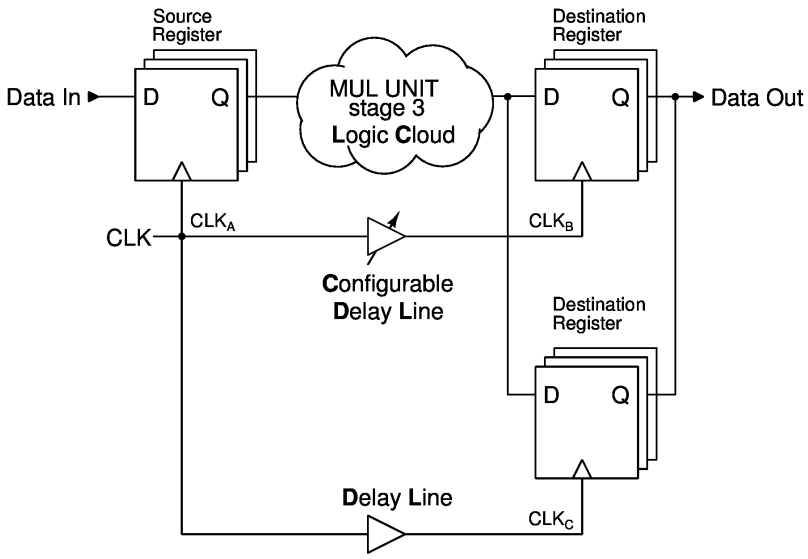

Figure 5.4 Mul-unit delay lines connections to endpoints.

As mentioned earlier, the Mul-unit is the last stage of a 32-bit multiplier circuit. The CDL was only inserted only to half of the endpoints of these Mul-units. The other half has a constant delay line and always runs at-speed. This is depicted in the simplified schematic in Fig. 5.4. To separate the results captured at different speeds, the SDD optimizer masks the uncontrollable endpoints when the test is not running at-speed.

5.2 Tester Setup

The tester that is used to test the AnARM chips is a Teradyne FLEX tester. In order to be able to test many AnARM chips on this tester, a special test fixture and printed circuit board (PCB) were designed. This PCB has a special socket that allows us to swap AnARM chips on a single PCB. The PCB contains voltage regulators to supply the chip with the needed voltage. The core of the AnARM run on 0.9V supply, while the I/Os require a 1.5V. The PCB also serves as an interface between the FLEX tester and the chip, where the appropriate connector (type QTH-90) is mounted on the back side of the PCB to connect to the FLEX tester high speed digital signals motherboard (has connectors of type QSH-90). Fig. 5.5 shows pictures of the test setup on the FLEX tester motherboard with the PCB, socket and AnARM chip.

On the software side, a converter is needed in order to convert the Tessent pattern files or the optimized SDD test files to pattern files that the FLEX tester can read and compile. An

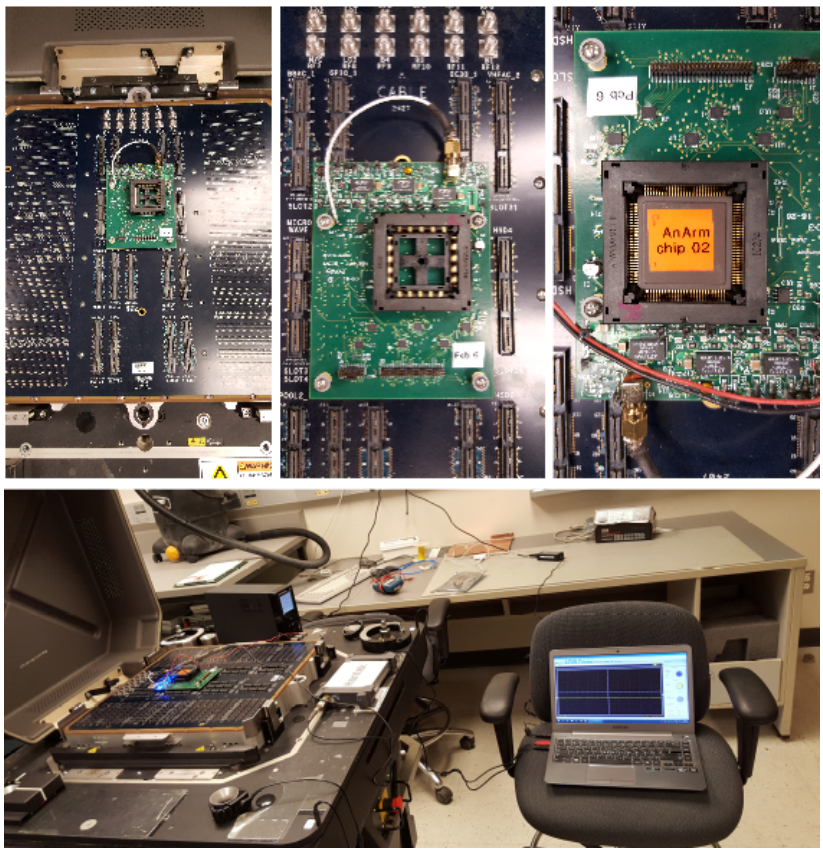

Figure 5.5 The test setup comprising of the FLEX tester with the PCB, socket and AnARM chip.

in-house converter, written in Python, was developed and tested to do the conversion. In the case of at-speed testing, the converter takes files written in WGL pattern format and converts them directly to the FLEX tester pattern format. However, to convert the optimized SDD test files into the FLEX tester format, more information is required (Fig. 5.6). The original pattern file in WGL format, the scan-cells order in the scan chains, as well as the mapping of the AnARM FI signals (CDL control bits) to the equivalent test clock speeds are needed to convert the optimized SDD test file into the FLEX tester patterns. The converter uses the WGL file to translate the pattern number in the optimized SDD file into the real pattern. It also uses the scan-cells order information to properly masks vectors in the patterns, and the clock speed mapping is used to set the FI bit in the pattern. One of the limitations of the available FLEX tester is that it has a limit on the number of applied patterns per file. For that reason, the pattern converter is equipped with the ability to split the output file into a set of smaller test patterns. These files are then applied to the chips in sequence.

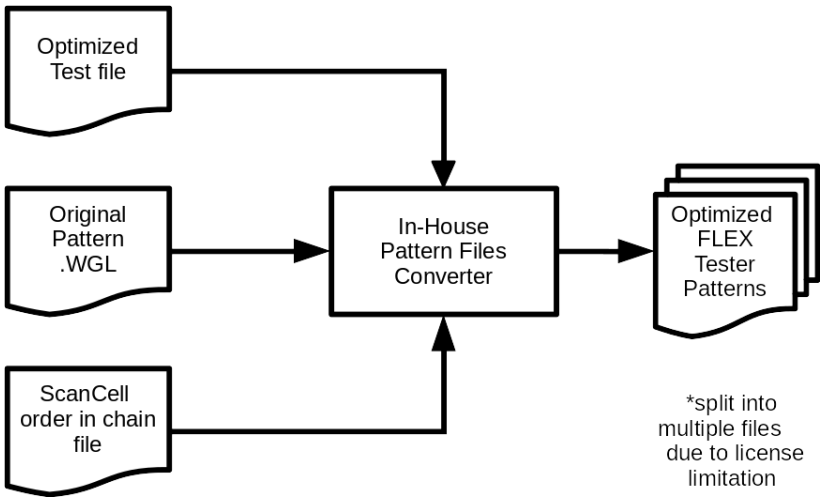

Figure 5.6 The pattern conversion process from the optimized SDD test file to the FLEX tester pattern

5.3 Test Sequence

The test of the AnARM chips goes through several steps. First, a test pattern is applied to read a known internal identification number inside the AnARM chips. This pattern is compared to both a correct and incorrect values in two test steps. This procedure is done at the beginning of the test to verify that the tester is properly configured to and that comparison of the data is done correctly. Then a scan chains test is applied to verify that the chains are fully functional and that they can shift the data from input to output without any errors. Once the scan chains are verified to work correctly, the AnARM chip is tested for stuck-at faults. Next, an at-speed delay test is applied to test for delay faults. The test is repeated while using faster test clock setting (by adjusting the FI bits of the CDL) to know if the new at-speed LOC method is working properly and to infer the available slack in the Mul-units. Here, the at-speed test is verified to be functioning correctly if the AnARM chip fails at higher test speed setting. Lastly, we apply the proposed optimized SDD test using the same pattern set of the at-speed test.

In the next sections, the results for the at-speed test and the optimized SDD test are presented. In this project, Fifty AnARM chips have been fabricated and packaged. All 50 chips have passed functional testing (done separately from this test sequence). However, out of those 50, 25 chips have failed the scan chain test (i.e. the scan chains are broken) and cannot be used for at-speed or optimized SDD testing. Thus, in the coming sections, only the results of the 25 AnARM chips with functional scan chains are discussed.

5.4 At-Speed Test Post-Silicon Results

For the 25 AnARM chips with functional scan chains, the at-speed test patterns were applied with 4 different test clock speeds: one at-speed test clock and three levels of faster-than-at-speed clocks. The test clock speed (i.e. the delay of the CDL) is configured by the FI signals that are controlled by two dedicated pins on the chip. Table 5.2 summarizes the test results by showing the percentage of passing and failing chips for the different test clock speeds. The chips start to fail the test when $FI_0FI_1 = 10$ (faster-than-at-speed level 2). In this case, each chip fails with a different number of failing endpoints. Table 5.3 shows the average number of failing endpoints for each chip under the faster-than-at-speed level 2 test. This average was obtained by repeating the at-speed test 11 times. This result gives us a hint on how much slack is available in each instance of the Mul-units. The higher the number of failing endpoints, the tighter the average slack. Since the slack is set by the delay of the CDL and Mul-units logic cloud, this could indicate that the chip is in faster process corner where the delay distribution is tighter. Some might wonder why the chips started to fail at level 2 of faster-than-at-speed and not at level 1. This is possibly because the patterns used are regular TDF patterns that are not generated by a timing aware engine. This implies that the ATPG algorithm selected short paths to test the Mul-units. Another possible explanation is that some of the very long paths that were listed in the timing analysis (Fig. 5.3) are false paths that cannot be sensitized. Lastly, the fact that increasing the speed of the test clock (CDL delay) failed the chips confirms that the applied LOC is working properly.

5.5 Optimized SDD Test Post-Silicon Results

This section presents the results of applying the proposed SDD test optimization technique on the Mul-units inside the AnARM chips. The Mul-units have a total of 56472 faults and 552 endpoints. The optimizer used the preexisting at-speed test structure and pattern set to generate the optimized SDD test. The applied at-speed pattern set has 202 patterns and achieves a TDFC of 96.07% and a WeSPer of 60.03%.

By applying the proposed SDD test optimization using the same pattern set as the at-speed test and the same set of test clock periods presented previously, we are able to increase the test quality (WeSPer value) to 69.71% if overtesting is allowed, and 68.92% if overtesting is prohibited. This is a relative increase of 16.12% and 14.8% in delay test quality. In these tests, the optimizer generated 30 test options per fault (i.e. a depth of 30 test options) and the grouping algorithm used a quality drop tolerance of 100%. Note that simulation trials, with

Table 5.2 Post-silicon test results of 25 AnARM chips at different test clock speeds

Test Speed	$\boldsymbol{FI_1FI_0}$	**Pass (%)**	**Fail (%)**
At-speed	"00"	100	0
Faster-than-at-speed level 1	"01"	100	0
Faster-than-at-speed level 2	"10"	24	76
Faster-than-at-speed level 3	"11"	0	100

Table 5.3 Number of failing endpoints when using faster-than-at-speed level 2 test speed

Chip ID	**Average Number of Failing Endpoints**	**Chip ID**	**Average Number of Failing Endpoints**
2	0	20	56.5
3	0	21	61
4	113	22	1.7
5	14.5	23	77.6
6	14.5	29	1
8	25.3	30	45
10	0	34	8
11	15	37	16
12	0	41	8.9
14	36.4	42	19.3
16	0	43	5.3
18	2.1	44	0
19	18.7	-	-

4 test clocks, have demonstrated to us that it is possible to obtain a $WeSPer_{opt}$ of 89.01% if the test clock periods were better selected. The final groups count for the optimized SDD test when overtesting is allowed is 796, and 808 when overtesting is prohibited. This is around a four folds increase in pattern count compared to the original pattern count and it is equivalent to applying the test once for each test clock speed. Table 5.4 summarizes the aforementioned results.

The optimized SDD test was applied on all 25 AnARM chips with functional scan chains. For each chip, the test was looped 11 times and the average results are listed in table 5.5.

Table 5.4 AnARM Mul-units and optimized SDD test characteristics

Number of Faults		56472
Number of Endpoints		552
Number of Patterns		202
TDFC		96.07%
At-Speed Test WeSPer		60.03%
	Overtest Prohibited	Overtest Allowed
$WeSPer_{max}$	69.11%	69.79%
TOPer	0.16%	0%
$WeSPer_{opt}$	68.92%	69.71%
Groups Count	796	808

Table 5.5 Test results of applying the optimized SDD test on the Mul-units

	Average Number of Failing Endpoints			**Average Number of Failing Endpoints**	
Chip ID #	**Overtest Allowed**	**Overtest Prohibited**	**Chip ID #**	**Overtest Allowed**	**Overtest Prohibited**
2	11.64	8.82	**20**	11.00	9.18
3	39.91	28.91	**21**	59.18	52.09
4	93.55	82.36	**22**	56.09	46.64
5	28.09	15.91	**23**	26.36	18.73
6	32.45	27.00	**29**	52.18	46.64
8	43.18	30.45	**30**	22.82	14.91
10	24.45	22.27	**34**	27.64	21.45
11	41.09	33.09	**37**	16.36	9.36
12	14.64	12.64	**41**	46.82	42.55
14	56.73	51.82	**42**	39.09	31.09
16	16.27	9.09	**43**	20.73	18.18
18	28.00	24.64	**44**	51.00	40.36
19	15.36	9.36	**-**	-	-

None of the 25 chips was able to fully pass the SDD test, however, the average number of failing endpoints differ from one chip to another. The results show that, for all chips, the average number of failing endpoints decrease when overtesting is avoided. To better visualize this, the results are sorted by the number of failing endpoints and plotted in Fig. 5.7. This proves that the algorithm avoids overtesting properly and shows the significance of avoiding overtesting the circuit. Nonetheless, remember that overtested faults that are caught in the

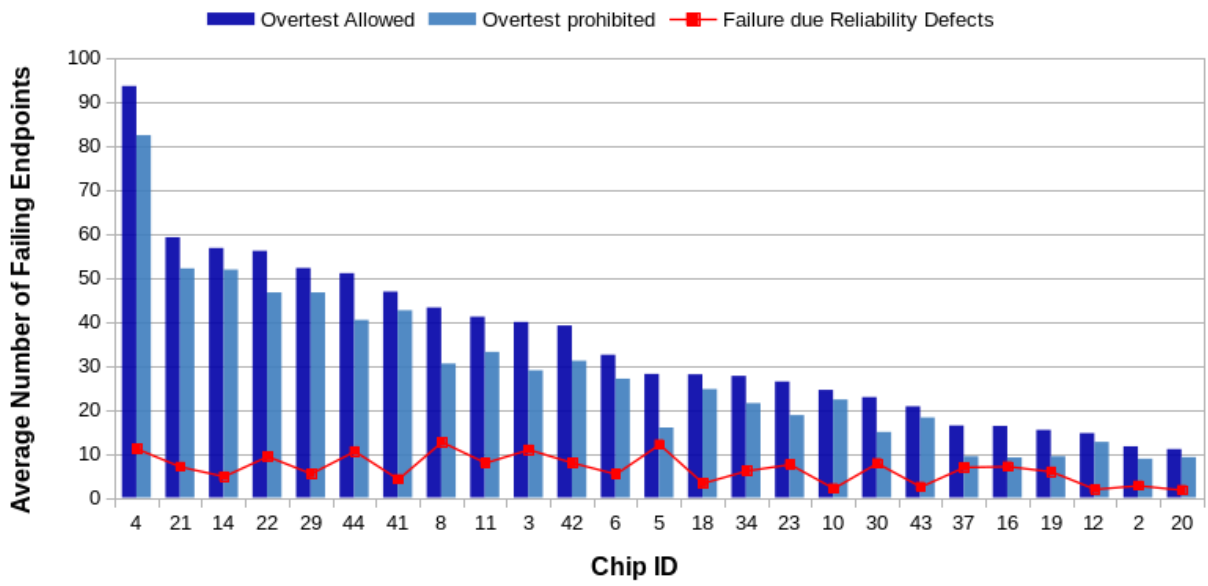

Figure 5.7 Optimized SDD test post-silicon test results.

optimized test are those with sizes close to the SEDD and could soon grow to fail the circuit under normal operation conditions.

Finally, we would like to point out that the test clock periods (i.e CDL delays) and the path delays were estimated by the timing tools based on the typical process corner at 0.9V and 25°C, and that the accuracy of the test results depends highly on the accuracy of this estimation. The estimation of the voltage and temperature is inferred from the test environment/setup. However, it is very difficult to estimate the process corner of the AnARM as it is a self-timed system. This means that part of the failing endpoints of AnARM chip, when testing the chips with the optimized SDD test, could be due improper estimation of the process corner. A method of estimating the post-fabrication process corner of an AnARM chip is outside the scope of this work and is recommended as part of the future work of this project.

CHAPTER 6 CONCLUSIONS AND RECOMMENDATIONS

6.1 Summary of Works

In this work we have developed a high quality delay test for a self-timed processor. The targeted processor is based on the Octasic self-timed architecture that uses a clocking mechanism based on configurable delay lines, and manages the access to resources with asynchronous control signals called tokens. This work is part of a bigger project where a testable self-timed processor that follows the Octasic architecture (named the AnARM) was developed by our research team. In this work, a high quality delay test was achieved by optimizing the usage of a preexisting at-speed test pattern set with the built-in CDLs to catch SDDs. The steps of developing the optimized SDD test, as well as the post-silicon test results, were presented in this dissertation.

In chapter 1, we discussed the importance of SDD testing and defined different types of delay faults and delay fault models. The basic functionality of the Octasic self-timed structure was also presented in chapter 1. In chapter 2, the most significant works in the literature on SDD testing were reviewed along with the prior work in this project (by other teammates) that concerns the development of a new LOC at-speed testing methodology for the AnARM structure.

In chapter 3, the development of a new high quality test for the AnARM started by defining an SDD metric that is more compatible with this architecture. Specifically, this chapter presented a novel SDD test model (the ideal SDD delay test model) and derived from it an SDD test quality metric (WeSPer) that enables us to take advantage of the CDLs built into the AnARM structure and create a high quality SDD delay test. WeSPer was formulated to be a flexible metric that can adapt to the availability of information on the test and the CUT by using CL multipliers. Since the goal is to test for effective SDDs, we formulated a CL multiplier for WeSPer that penalizes overtesting. With this multiplier, WeSPer is able to report the delay test quality for all ranges of test clock speed (slow, at-speed, faster-than-at-speed) without saturating or reporting illogical values. WeSPer is mainly a non-statistical SDD test quality metric, however, in case an accurate delay defect distribution is available, a CL multipliers that can take into account that distribution was developed. This work also included a debugging metric (named TOPer) to help understand how overtesting affected a test, if needed.

WeSPer was validated in chapter 3 with an extensive 54 PVT point simulation of the 28nm FD-SOI technology from STMicroelectronics on a set of ISCAS-85, 74X-Series and ITC99 benchmark circuits. From the simulation results, the sensitivity of WeSPer and other SDD test quality metrics to the test clock speed and PVT point change were discussed. WeSPer was shown to be the most accurately sensitive metric to the change of test-escape (or overtest) window. Moreover, two methods of estimating WeSPer over 54 PVT points were discussed. The first was by using 3 PVT points and fitting the results on an exponential curve. The fitting curve achieved an average percentage of error $< 1.5\%$ for all benchmark circuits. The second method was by finding the worst case scenario. We showed that the worst case would be at the slowest process, lowest voltage and lowest temperature. Finally we demonstrated that adding a delay defect distribution to WeSPer only changed the value by less than 5%.

The main optimized SDD test was developed in chapter 4. The proposed SDD test optimization method maximized the quality of the test (measured by WeSPer) using a set of predefined test clock speeds and TDF test patterns. For each fault in the CUT, the proposed method searches for the pattern/test-clock pair that would maximize the WeSPer value, then it adds the proper masking vector to avoid false failures. Two types of SDD test optimization were explored. The first only targeted effective SDDs, whereas the second also included reliability defects that are close to failing the circuit. In order to minimize the test time, a fault grouping algorithm that can reduce the final pattern count while preserving the SDD test quality was also discussed. The masking strategy was carefully developed so that it would not have a significant effect on fault grouping. This SDD test optimization method can be applied to the AnARM structure, as well as synchronous systems that are compatible with faster-than-at-speed testing. The proposed optimization technique was analyzed and verified on a set of selected benchmark circuits implemented in the 28nm FD-SOI CMOS process technology. The benchmark circuits results showed that the proposed optimization and grouping techniques are effective and work as expected. The set of results have shown that increasing the number of test clocks used or changing the type of pattern set (ATPG or TAA) only resulted in a slight benefit. In the case of increasing the number of test clocks used, the groups count (final pattern count) also increased. Lastly, we noticed that avoiding overtesting during optimization resulted in a slight reduction in the optimized SDD test quality.

The newly developed optimized SDD test was finally applied on the fabricated AnARM chips and the results were presented in chapter 5. 50 AnARM chips were fabricated in the 28nm FD-SOI technology from STMicroelectronics. Before applying the proposed optimized

test, the effectiveness of the previously developed at-speed test was verified along with the new AnARM LOC strategy. A Teradyne FLEX tester was used to apply the developed tests to the AnARM chips. The test procedure started with verifying the correctness of the test setup and the functionality of the scan-chains that are build inside the AnARM. The 25 AnARM chips that passed the scan-chains test were then at-speed tested and the results were presented. All of these 25 chips passed the regular at-speed test. The test was repeated 3 more times with 3 levels of faster-than-at-speed test clocks and the results showed that the AnARM chips start to fail at level 2. Since the test was able to catch these failures, we concluded that the at-speed test is functioning correctly.

Next, the optimizer used the same test patterns as the at-speed test with the existing built-in test clock speeds of the AnARM to build an optimized SDD test. With 30 test options per fault and 100% quality drop tolerance, the optimizer was able to build a delay test that improves the delay test quality (measured by WeSPer) by up to 16.12% (from 60.03% to 69.71%). Moreover, trials with the optimizers have shown that a WeSPer_{opt} value of 89.01% is achievable with 4 test clocks if the CDL delays were better selected. The final pattern count of the optimized SDD test increased by no more than 4 times the initial at-speed test pattern count. Two optimized SDD tests were applied to the AnARM chips. One that allowed overtesting (only if the defect is close to failing the circuit) and the other that completely avoided overtesting. None of the 25 AnARM chips were able to pass the optimized SDD test, however, the number of failing endpoints for each chip was different. The average number of failing endpoints varied between 93.55 and 8.82. The test that prohibited overtesting had a lower number of failing endpoints for each chip compared to the test that allowed overtesting. This proves that the optimized test is working properly. Finally note that the accuracy of the optimized SDD test results depends highly of the estimated path delays with respect to the fabricated chip.

6.2 Limitations

In this work there are two sources of limitations. The first limitation comes from the fact that the WeSPer computation flow relies on an industrial tool to extract the delay tables and the delay information of the paths. This means that the results of this work is prone to any approximation that the tool makes. For example, during the generation of the path delay tables, sometimes the longest path delay reported for a fault was found to be shorter than the one activated by one of the patterns. In this case, the longest path delay reported for that fault is corrected by simply setting it to the highest delay value found for that fault.

In addition, since the structural delay information is not reported by the tool, no analysis on cross-talk or path delay distribution can be used to enhance the test accuracy. Moreover, longer simulation times were needed to extract the delay tables from the industrial tool since the tool only reports the longest activated path for each fault over all endpoints. Recall that the WeSPer calculation required the delay longest path delay for each fault to each endpoint. Also, the activated path delay tables could be large for larger circuits. This means that they can take large space in the memory.

The other limitation, that is related to the self-timed nature of the AnARM, stems from the fact that the exact test clock speed (i.e. CDL delay) after fabrication is unknown. Since there is no PLL that can correct the test clock speed based on an external signal, the CDLs delays are highly a function of PVT. The accuracy of the proposed optimized SDD test depends on the accuracy of the delay information used. That being said, as long as the delay of the CDL and the delay of the tested logic cloud change by the same amount with PVT changes, the test results will stay valid.

6.3 Future Work and Possible Improvements

In light of the previously discussed limitations, several improvements could be suggested. Firstly, the delay table extraction speeds can be significantly reduced if a dedicated true path delay timing tool is developed specifically for WeSPer needs, or if access to fast true path timing tools, such as PHEATON [76], can be obtained. This will also help to add structural information to WeSPer by developing appropriate CL multipliers. Also, the activated path delay tables could be compressed to remove empty entries and maybe a memory efficient representation could be found. Secondly, a method of estimating the slack in the post-silicon chip can be developed. If this information is fed back to the optimizer, the generated test is deemed to be more accurate. Moreover, WeSPer can be further enhanced by formulating a CL multiplier that can include the uncertainty of extracted delay information due to the PVT variations. This can be done by running a multi-PVT-point simulation and recording the variations that happen at each data point in the delay table. Lastly, developing an algorithm to determine the test clock speeds that can maximize the optimized SDD test based on the extracted delay tables of the CUT is a logical next step for this project.

REFERENCES

[1] M. Laurence, "Introduction to octasic asynchronous processor technology," in *Asynchronous Circuits and Systems (ASYNC), 2012 18th IEEE International Symposium on*, 2012, pp. 113–117. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6243889

[2] P. Gronowski *et al.*, "High-performance microprocessor design," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 5, pp. 676–686, 1998. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=668981

[3] A. Yakovlev, P. Vivet, and M. Renaudin, "Advances in asynchronous logic: From principles to gals noc, recent industry applications, and commercial cad tools," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2013, pp. 1715–1724.

[4] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective (4th Edition)*. Pearson, 2010. [Online]. Available: http://www.amazon.com/CMOS-VLSI-Design-Circuits-Perspective/dp/0321547748%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0321547748

[5] S. K. Goel, *Testing for Small-Delay Defects in Nanoscale CMOS Integrated Circuits (Devices, Circuits, and Systems)*. CRC Press, 2013. [Online]. Available: http://www.amazon.com/Testing-Small-Delay-Nanoscale-Integrated-Circuits-ebook/dp/B00FZHV61K%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3DB00FZHV61K

[6] X. Zhang, K. Xiao, and M. Tehranipoor, "Path-delay fingerprinting for identification of recovered ics," in *2012 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct 2012, pp. 13–18.

[7] M. Ikeda *et al.*, "Datapath delay distributions for data/instruction against pvt variations in 90nm cmos," in *2007 14th IEEE International Conference on Electronics, Circuits and Systems*, Dec 2007, pp. 154–157.

[8] K. C. Mohammad Tehranipoor, Ke Peng, *Test and Diagnosis for Small-Delay Defects*. Springer New York, 2011. [Online]. Avail-

able: http://www.ebook.de/de/product/18536045/mohammad_tehranipoor_ke_peng_krishnendu_chakrabarty_test_and_diagnosis_for_small_delay_defects.html

[9] T. C. Chen *et al.*, "E-beam inspection for gap physical defect detection in 28nm cmos process," in *ASMC 2013 SEMI Advanced Semiconductor Manufacturing Conference*, May 2013, pp. 307–309.

[10] W. P. Peng *et al.*, "Reduction of "dark gate" defects in replacement-metal-gate process and middle-of-line contacts for advanced planar cmos and finfet technology," in *2016 China Semiconductor Technology International Conference (CSTIC)*, March 2016, pp. 1–4.

[11] S. Zhong *et al.*, "Analysis of resistive bridge defect delay behavior in the presence of process variation," in *2011 Asian Test Symposium*, Nov 2011, pp. 389–394.

[12] G. L. Smith, "Model for delay faults based upon paths." in *ITC*, 1985, pp. 342–351.

[13] J. Waicukauski *et al.*, "Transition fault simulation," *IEEE Des Test Comput*, vol. 4, no. 2, pp. 32–38, 1987. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4069962

[14] C. J. Lin and S. Reddy, "On delay fault testing in logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 6, no. 5, pp. 694–703, 1987. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1270315

[15] S. Devadas and K. Keutzer, "Validatable nonrobust delay-fault testable circuits via logic synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 12, pp. 1559–1573, 1992. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=180267

[16] K.-T. Cheng and H.-C. Chen, "Classification and identification of nonrobust untestable path delay faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 8, pp. 845–853, 1996. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=511566

[17] X. Lin *et al.*, "Timing-aware ATPG for high quality at-speed testing of small delay defects," in *Test Symposium, 2006. ATS '06. 15th Asian*, 2006, pp. 139–146. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4030760

[18] I. Pomeranz and S. M. Reddy, "An efficient nonenumerative method to estimate the path delay fault coverage in combinational circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 13, no. 2, pp. 240–250, 1994. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=259947

[19] I. Pomeranz and S. Reddy, "Transition path delay faults: A new path delay fault model for small and large delay defects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 1, pp. 98–107, 2008. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4374122

[20] Y. Sato *et al.*, "Invisible delay quality-sdqm model lights up what could not be seen," in *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*. IEEE, 2005, pp. 9–pp. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1584088

[21] M. T. Mohammadat *et al.*, "Multivoltage aware resistive open fault model," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 2, pp. 220–231, Feb. 2014.

[22] N. Ahmed, M. Tehranipoor, and V. Jayaram, "Supply voltage noise aware ATPG for transition delay faults," in *VLSI Test Symposium, 2007. 25th IEEE*, 2007, pp. 179–186. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4209909

[23] F. Bao, M. Tehranippor, and H. Chen, "Worst-case critical-path delay analysis considering power-supply noise," in *Test Symposium (ATS), 2013 22nd Asian*, 2013, pp. 37–42. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6690611

[24] S. Pant *et al.*, "Vectorless analysis of supply noise induced delay variation," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, 2003, pp. 184–191. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1257629

[25] K. Peng *et al.*, "Crosstalk- and process variations-aware high-quality tests for small-delay defects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 6, pp. 1129–1142, 2013. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6262496

[26] B. Kruseman *et al.*, "On hazard-free patterns for fine-delay fault testing," in *Test Conference, 2004. Proceedings. ITC 2004. International*, 2004, pp. 213–222. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1386955

[27] M. Bushnell and V. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits (Frontiers in Electronic Testing)*. Springer, 2005. [Online]. Available: http://www.amazon.com/Essentials-Electronic-Mixed-Signal-Circuits-Frontiers/dp/0792379918%3FSubscriptionId%3D0JYN1NVW651KCA56C102%26tag%3Dtechkie-20%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0792379918

[28] J. Savir, "Skewed-load transition test: Part i, calculus," in *Test Conference, 1992. Proceedings., International*, 1992. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=527892

[29] J. Savir and S. Patil, "Broad-side delay test," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 8, pp. 1057–1064, 1994. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=298042

[30] T. Awad *et al.*, "Method for sharing a resource and circuit making use of same," U.S. Patent 8 689 218, Apr. 1, 2014. [Online]. Available: https://www.google.com/patents/US8689218

[31] ——, "Clock signal propagation method for integrated circuits (ics) and integrated circuit making use of same," U.S. Patent 8 130 019, Mar. 6, 2012. [Online]. Available: https://www.google.com/patents/US8130019

[32] M. Fiorentino *et al.*, "Self-timed circuits fpga implementation flow," in *New Circuits and Systems Conference (NEWCAS), 2015 IEEE 13th International*, June 2015, pp. 1–4.

[33] N. Ahmed, M. Tehranipoor, and V. Jayaram, "Timing-based delay test for screening small delay defects," in *Proceedings of the 43rd annual Design Automation Conference*. ACM, 2006, pp. 320–325. [Online]. Available: http://dl.acm.org/citation.cfm?id=1146993

[34] F. Bao *et al.*, "Generation of effective 1-Detect tdf patterns for detecting small-delay defects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 10, pp. 1583–1594, 2013. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6601027

[35] C.-Y. Chang *et al.*, "Compact test pattern selection for small delay defect," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32,

no. 6, pp. 971–975, 2013. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6516596

[36] S. K. Goel, N. Devta-Prasanna, and R. P. Turakhia, "Effective and efficient test pattern generation for small delay defect," in *VLSI Test Symposium, 2009. VTS'09. 27th IEEE*. IEEE, 2009, pp. 111–116. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5116618

[37] P. Gupta and M. S. Hsiao, "Alaptf: A new transition fault model and the atpg algorithm," in *Test Conference, 2004. Proceedings. ITC 2004. International*. IEEE, 2004, pp. 1053–1060. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1387378

[38] K. Peng *et al.*, "A novel hybrid method for sdd pattern grading and selection," in *VLSI Test Symposium (VTS), 2010 28th*, 2010, pp. 45–50. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5469619

[39] ——, "High-quality pattern selection for screening small-delay defects considering process variations and crosstalk," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, 2010, pp. 1426–1431. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5457036

[40] W. Qiu *et al.*, "K longest paths per gate (klpg) test generation for scan-based sequential circuits," in *Test Conference, 2004. Proceedings. ITC 2004. International*, 2004, pp. 223–231. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1386956

[41] R. Tayade, S. Sundereswaran, and J. Abraham, "Small-delay defect detection in the presence of process variations," in *Quality Electronic Design, 2007. ISQED '07. 8th International Symposium on*, 2007, pp. 711–716. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4149118

[42] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-pattern selection for screening small-delay defects in very-deep submicrometer integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 5, pp. 760–773, 2010. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5452121

[43] ——, "Test-pattern grading and pattern selection for small-delay defects," in *VLSI Test Symposium, 2008. VTS 2008. 26th IEEE*, 2008, pp. 233–239. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4511728

[44] D. Xu *et al.*, "Test-quality optimization for variable -detections of transition faults," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 8, pp. 1738–1749, Aug. 2014.

[45] X. Fu, H. Li, and X. Li, "Testable path selection and grouping for faster than at-speed testing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 2, pp. 236–247, Feb 2012.

[46] S. Pei *et al.*, "An on-chip frequency programmable test clock generation and application method for small delay defect detection," vol. 49, pp. 87 – 97. [Online]. Available: //www.sciencedirect.com/science/article/pii/S0167926014000893

[47] R. Tayade and J. A. Abraham, "On-chip programmable capture for accurate path delay test and characterization," in *Test Conference, 2008. ITC 2008. IEEE International*. IEEE, 2008, pp. 1–10. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4700564

[48] S. Pei *et al.*, "Enhanced lccg: A novel test clock generation scheme for faster-than-at-speed delay testing," in *The 20th Asia and South Pacific Design Automation Conference*, Jan 2015, pp. 514–519.

[49] N. Ahmed and M. Tehranipoor, "A novel faster-than-at-speed transition-delay test method considering ir-drop effects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1573–1582, Oct 2009.

[50] M. Kampmann and S. Hellebrand, "Design-for-fast: Supporting x-tolerant compaction during faster-than-at-speed test," in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, April 2017, pp. 35–41.

[51] S. Chakravarty *et al.*, "Silicon evaluation of faster than at-speed transition delay tests," in *2012 IEEE 30th VLSI Test Symposium (VTS)*, April 2012, pp. 80–85.

[52] T. Yoneda *et al.*, "Faster-than-at-speed test for increased test quality and in-field reliability," in *2011 IEEE International Test Conference*, Sept 2011, pp. 1–9.

[53] F. Bao *et al.*, "Critical fault-based pattern generation for screening sdds," in *European Test Symposium (ETS), 2011 16th IEEE*, 2011, pp. 177–182. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5957943

[54] R. Tayade and J. Abraham, "Small-delay defect detection in the presence of process variations," *Microelectronics Journal*, vol. 39, no. 8, pp. 1093 –

1100, 2008, european Nano Systems (ENS) 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0026269208000098

[55] B. Arslan and A. Orailoglu, "Full exploitation of process variation space for continuous delivery of optimal delay test quality," in *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, 2013, pp. 552–557. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6509654

[56] F. Galarza-Medina *et al.*, "Small-delay defects detection under process variation using inter-path correlation," in *VLSI Test Symposium (VTS), 2012 IEEE 30th*, 2012, pp. 127–132. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6231091

[57] M. Kampmann *et al.*, "Optimized selection of frequencies for faster-than-at-speed test," in *2015 IEEE 24th Asian Test Symposium (ATS)*, Nov 2015, pp. 109–114.

[58] ——, "Built-in test for hidden delay faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2018.

[59] L. Shi and X. Cai, "An exact fast algorithm for minimum hitting set," in *2010 Third International Joint Conference on Computational Science and Optimization*, vol. 1, May 2010, pp. 64–67.

[60] S. Jin *et al.*, "Unified capture scheme for small delay defect detection and aging prediction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 5, pp. 821–833, 2013. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6243222

[61] N. Tendolkar, "Analysis of timing failures due to random AC defects in VLSI modules," in *Design Automation, 1985. 22nd Conference on*, 1985, pp. 709–714. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1586020

[62] N. Devta-Prasanna *et al.*, "Accurate measurement of small delay defect coverage of test patterns," in *Test Conference, 2009. ITC 2009. International*, 2009, pp. 1–10. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5355644

[63] V. Iyengar, B. K. Rosen, and I. Spillinger, "Delay test generation. i. concepts and coverage metrics," in *Test Conference, 1988. Proceedings. New Frontiers in Testing, International*, 1988, pp. 857–866. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=207873

[64] E. S. Park, M. Mercer, and T. Williams, "The total delay fault model and statistical delay fault coverage," *IEEE Transactions on Computers*, vol. 41, no. 6, pp. 688–698, 1992. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=144621

[65] S. Mitra *et al.*, "Delay defect screening using process monitor structures," in *2013 IEEE 31st VLSI Test Symposium (VTS)*. IEEE Computer Society, 2004, pp. 43–43. [Online]. Available: http://www.computer.org/csdl/proceedings/vts/2004/2134/00/21340043.pdf

[66] O. A. T. Hasib, Y. Savaria, and C. Thibeault, "Wesper: A flexible small delay defect quality metric," in *Proc. IEEE 34th VLSI Test Symp. (VTS)*, Apr. 2016, pp. 1–6.

[67] F. Shi and Y. Makris, "Testing delay faults in asynchronous handshake circuits," in *2006 IEEE/ACM International Conference on Computer Aided Design*, Nov 2006, pp. 193–197.

[68] G. Gill *et al.*, "Low-overhead testing of delay faults in high-speed asynchronous pipelines," in *12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'06)*, March 2006, pp. 11 pp.–56.

[69] F. te Beest and A. Peeters, "A multiplexer based test method for self-timed circuits," in *11th IEEE International Symposium on Asynchronous Circuits and Systems*. IEEE, March 2005, pp. 166–175.

[70] F. T. Beest *et al.*, "Automatic scan insertion and test generation for asynchronous circuits," in *Proceedings. International Test Conference*. IEEE, 2002, pp. 804–813.

[71] X. Lin *et al.*, "High-frequency, at-speed scan testing," *IEEE Design Test of Computers*, vol. 20, no. 5, pp. 17–25, Sept 2003.

[72] O. A. Hasib *et al.*, "Exploiting built-in delay lines for applying launch-on-capture at-speed testing on self-timed circuits," in *2018 IEEE 36th VLSI Test Symposium (VTS)*, April 2018, pp. 1–6.

[73] Synopsys, "Design compiler." [Online]. Available: https://www.synopsys.com/support/training/rtl-synthesis/design-compiler-rtl-synthesis.html

[74] Mentor, "The tessent product suit." [Online]. Available: https://www.mentor.com/products/silicon-yield/tessent/

[75] I. Pomeranz and S. M. Reddy, "Path selection for transition path delay faults," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 3, pp. 401–409, March 2010.

[76] M. Sauer, B. Becker, and I. Polian, "Phaeton: A sat-based framework for timing-aware path sensitization," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1869–1881, June 2016.

[77] M. Wagner and H. J. Wunderlich, "Probabilistic sensitization analysis for variation-aware path delay fault test evaluation," in *2017 22nd IEEE European Test Symposium (ETS)*, May 2017, pp. 1–6.

[78] A. Dasdan and I. Hom, "Handling inverted temperature dependence in static timing analysis," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 11, no. 2, pp. 306–324, Apr. 2006. [Online]. Available: http://doi.acm.org/10.1145/1142155.1142158

APPENDIX A ADDITIONAL METRIC SIMULATION RESULTS

In this appendix, the results that were summarized in chapter 3 are expanded. These result figures are listed here only for reference. The metric results in these figures behave similar to Fig. 3.4, and thus, they lead to the same conclusions. Two sets of figures for each benchmark circuits are listed below: one for the ATPG pattern set, and the other for a TAA pattern set. Remeber, the metrics are computed with 3 test clock speeds: slow, at-speed and fast. The slow clock period is 10% slower than the system clock speed, the at-speed test clock period is equal to the system clock period, and the fast test clock period is 10% faster that the system clock speed. As these results are part of the multi-PVT simulations, the system clock period here is set to be 25% larger than the longest path delay of the CUT under the slowest PVT point.

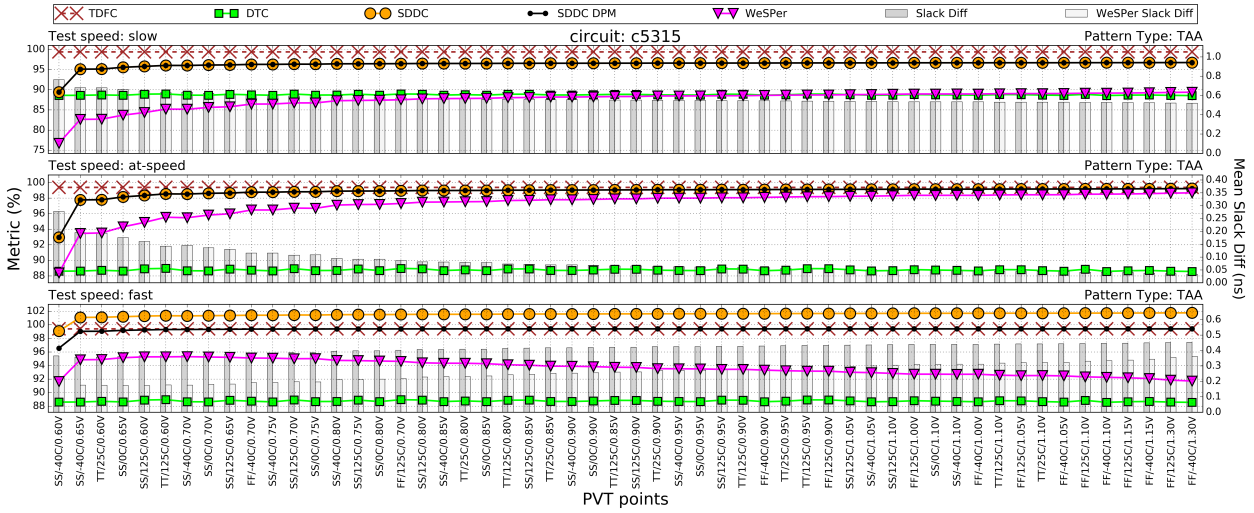

Figure A.1 Metric results for all available PVT points for the c5315. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

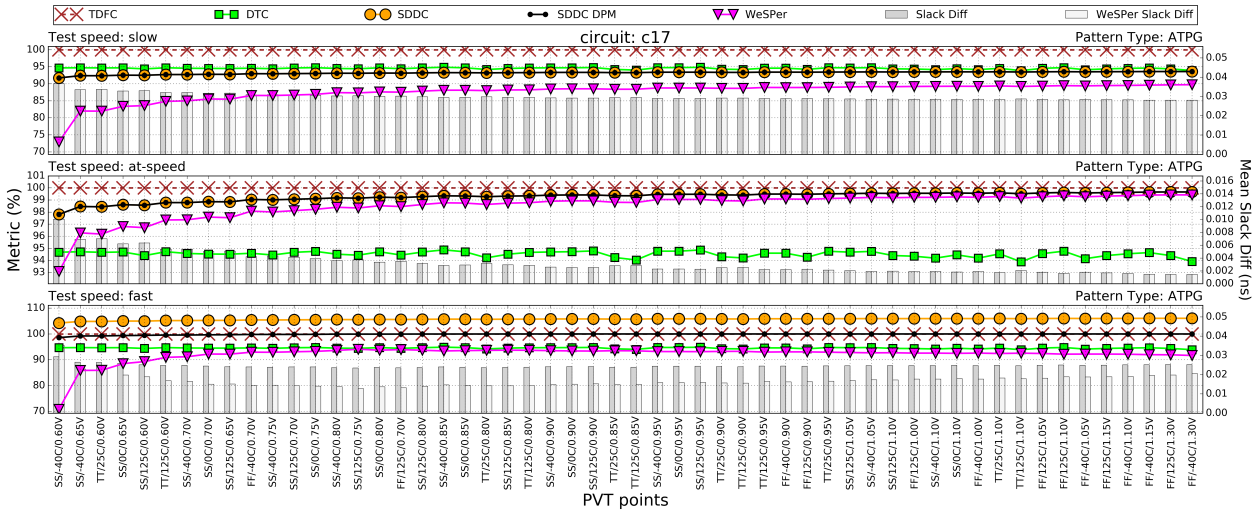

Figure A.2 Metric results for all available PVT points for the c17. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

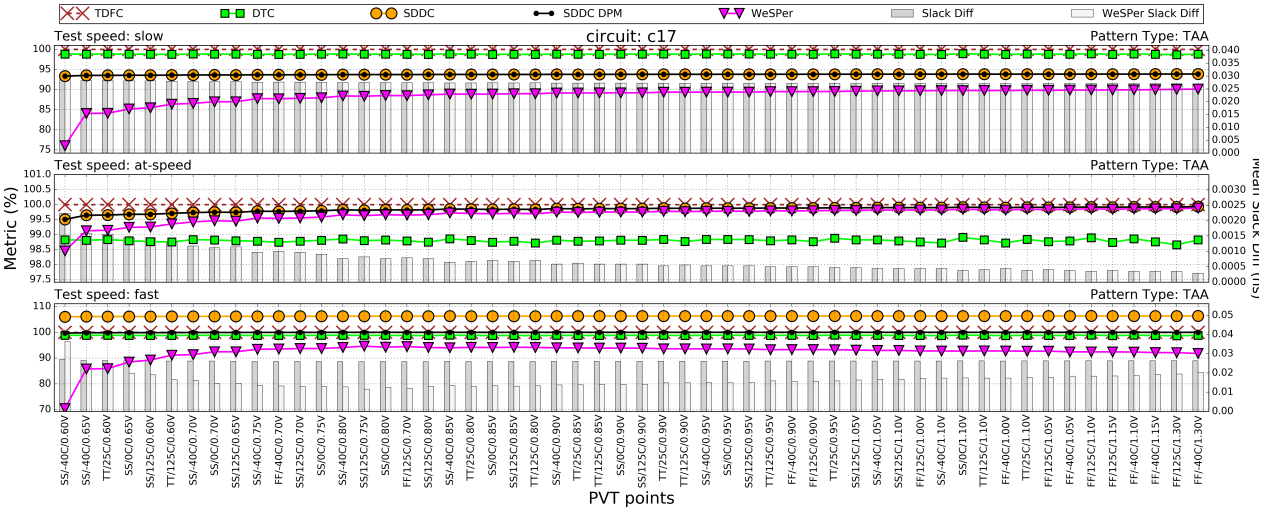

Figure A.3 Metric results for all available PVT points for the c17. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

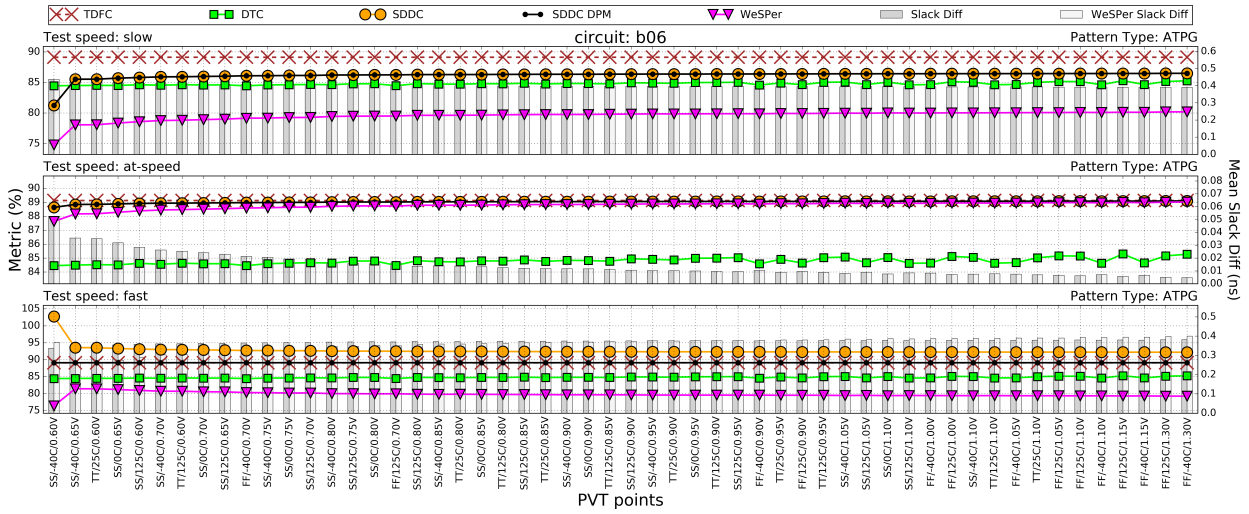

Figure A.4 Metric results for all available PVT points for the b06. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

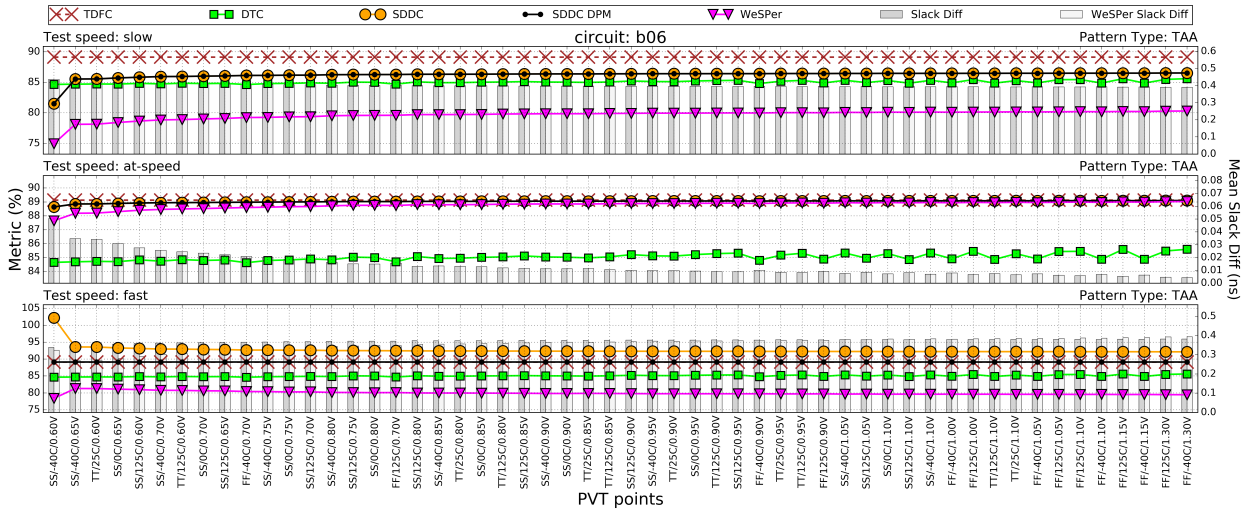

Figure A.5 Metric results for all available PVT points for the b06. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

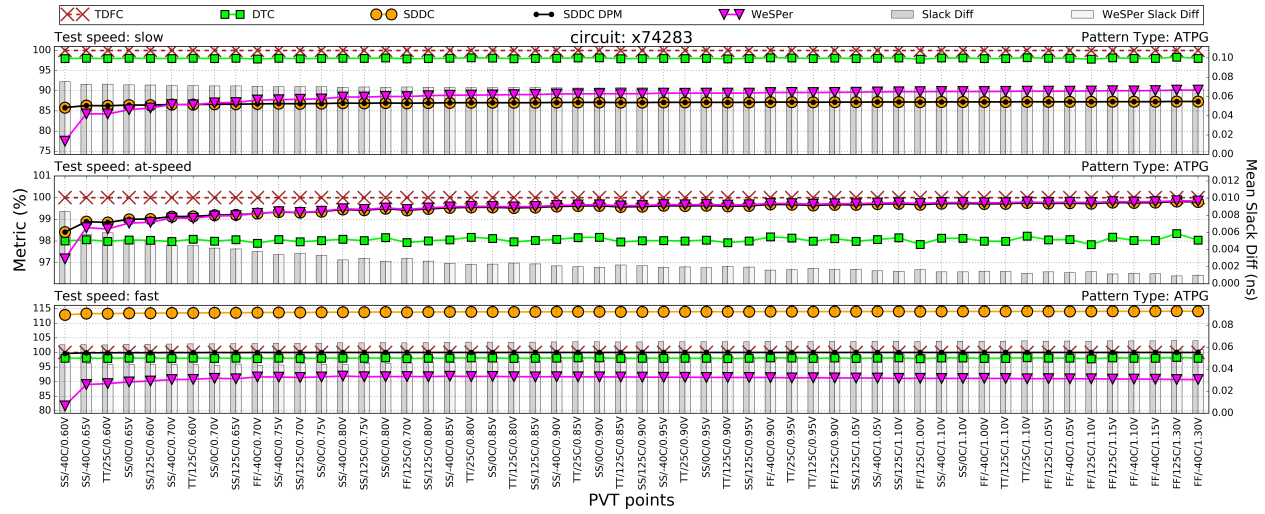

Figure A.6 Metric results for all available PVT points for the x74283. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

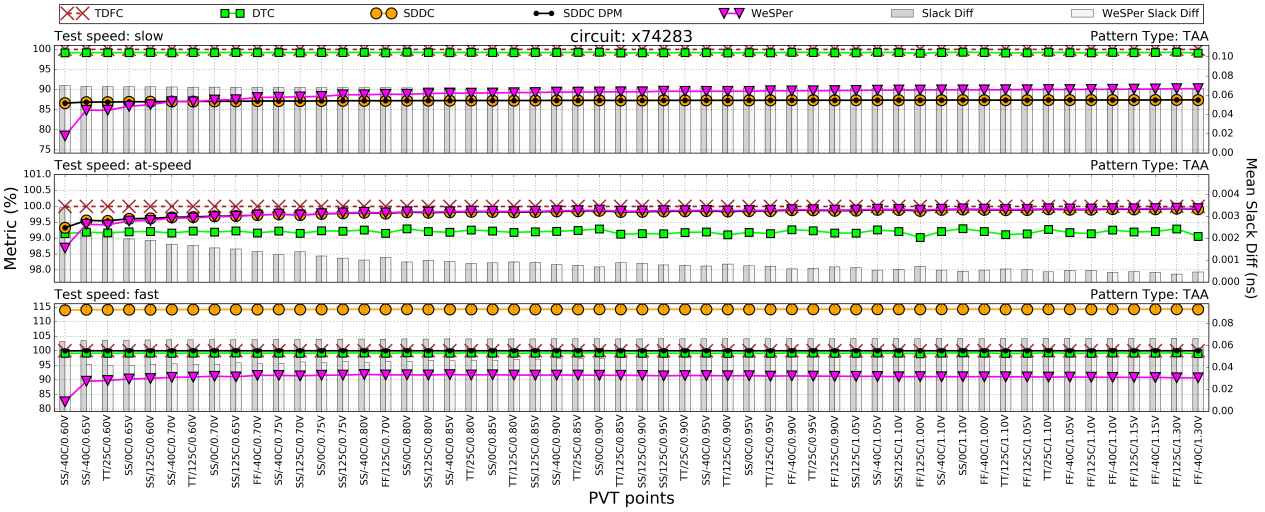

Figure A.7 Metric results for all available PVT points for the x74283. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

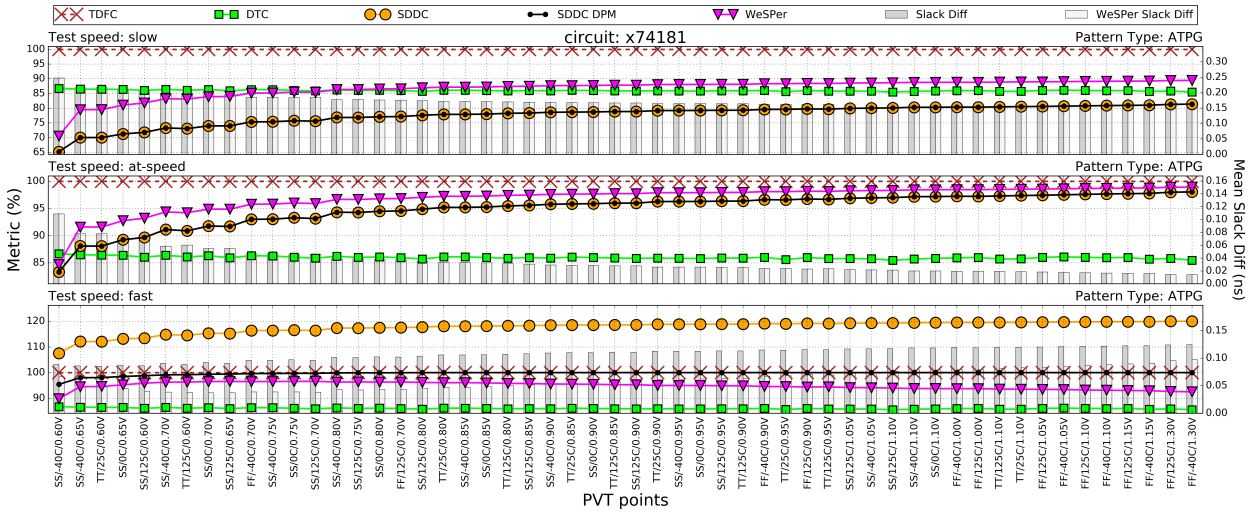

Figure A.8 Metric results for all available PVT points for the x74181. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

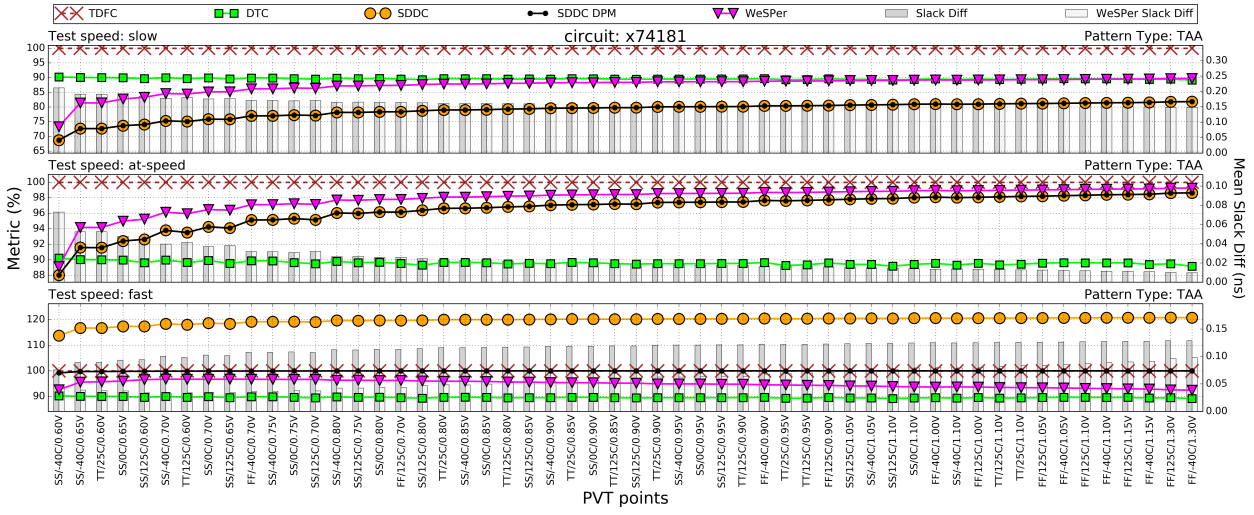

Figure A.9 Metric results for all available PVT points for the x74181. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

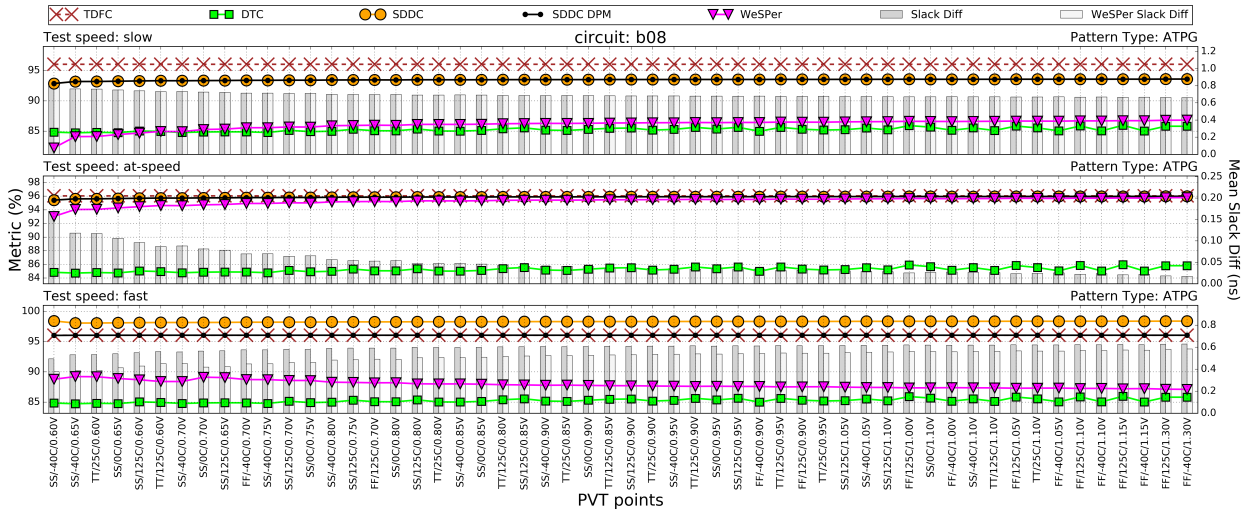

Figure A.10 Metric results for all available PVT points for the b08. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

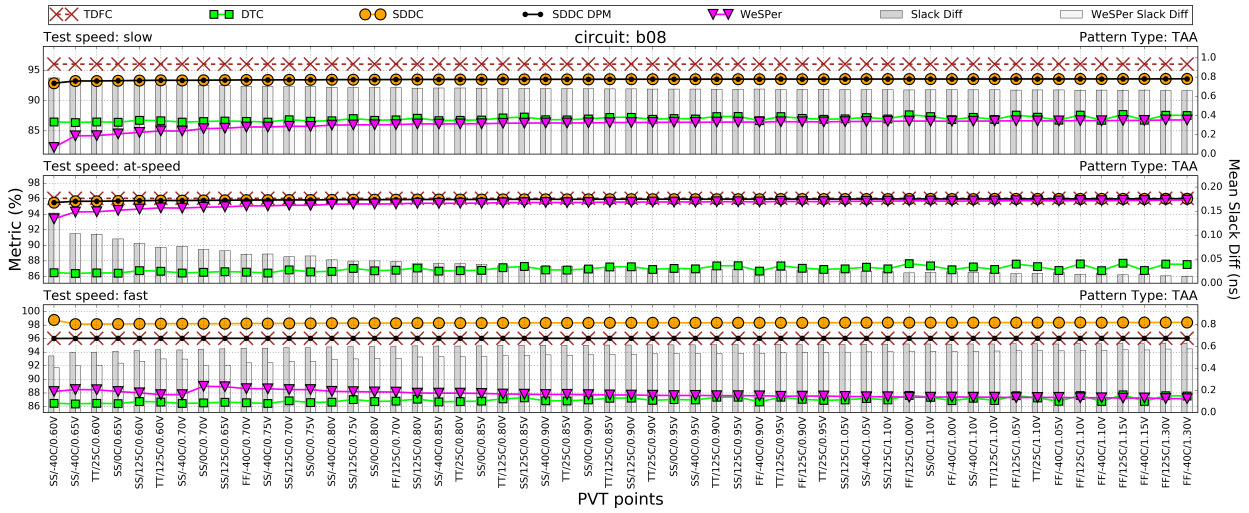

Figure A.11 Metric results for all available PVT points for the b08. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

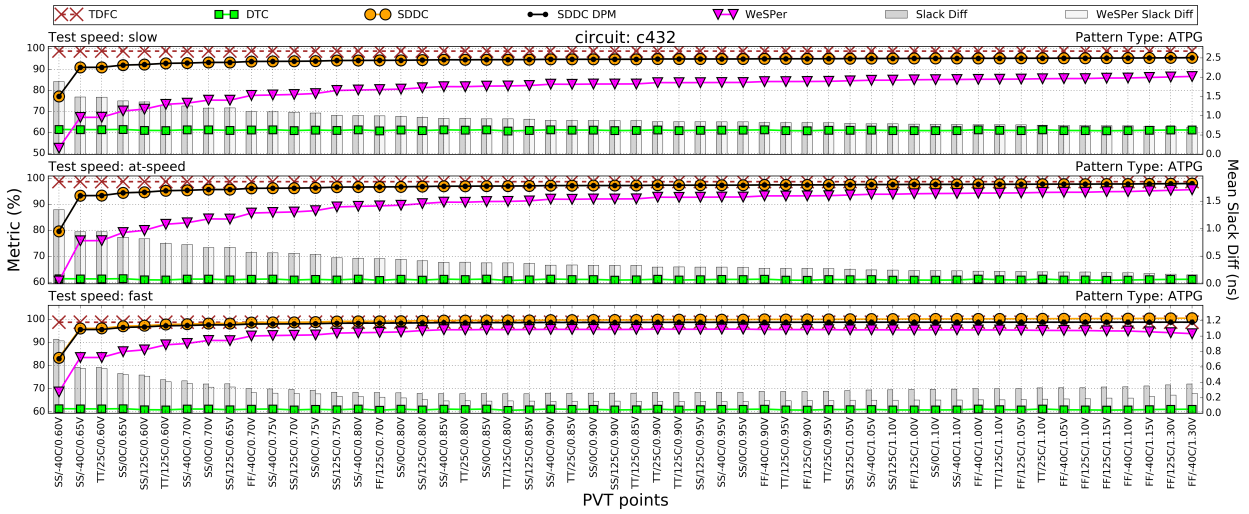

Figure A.12 Metric results for all available PVT points for the c432. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

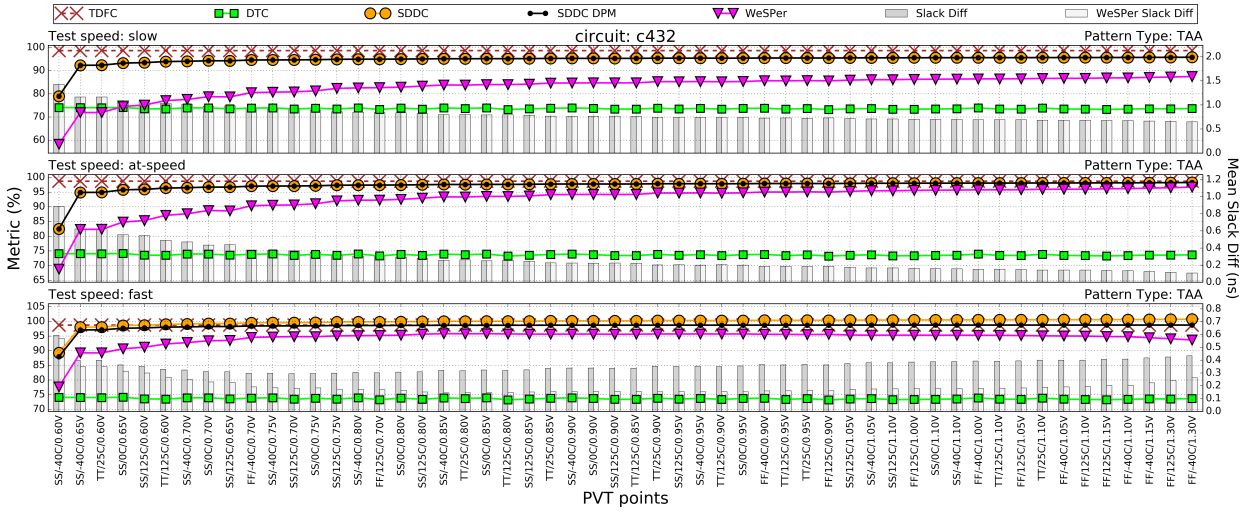

Figure A.13 Metric results for all available PVT points for the c432. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

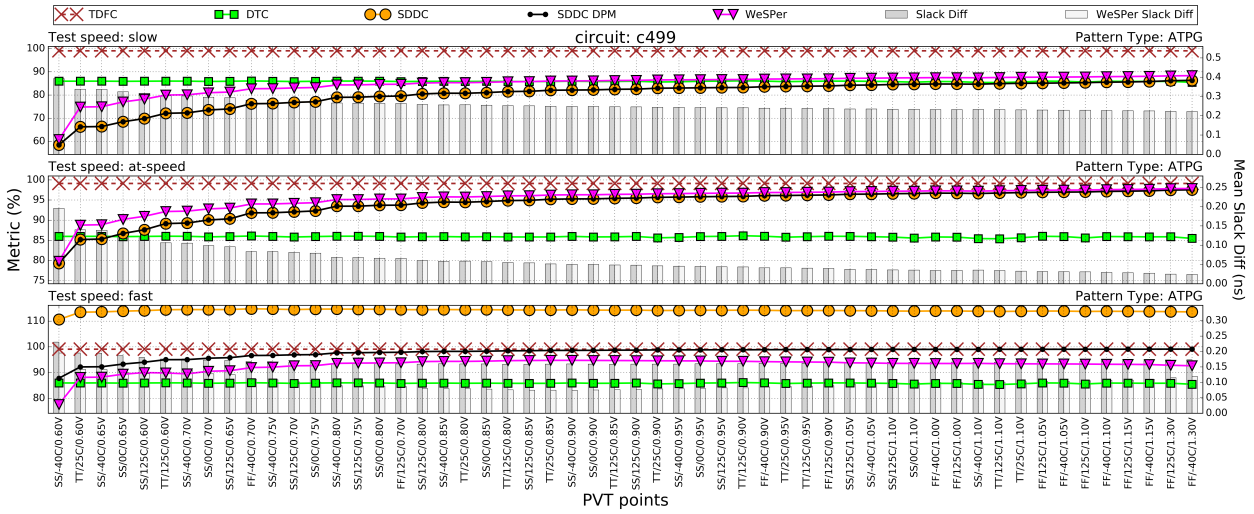

Figure A.14 Metric results for all available PVT points for the c499. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

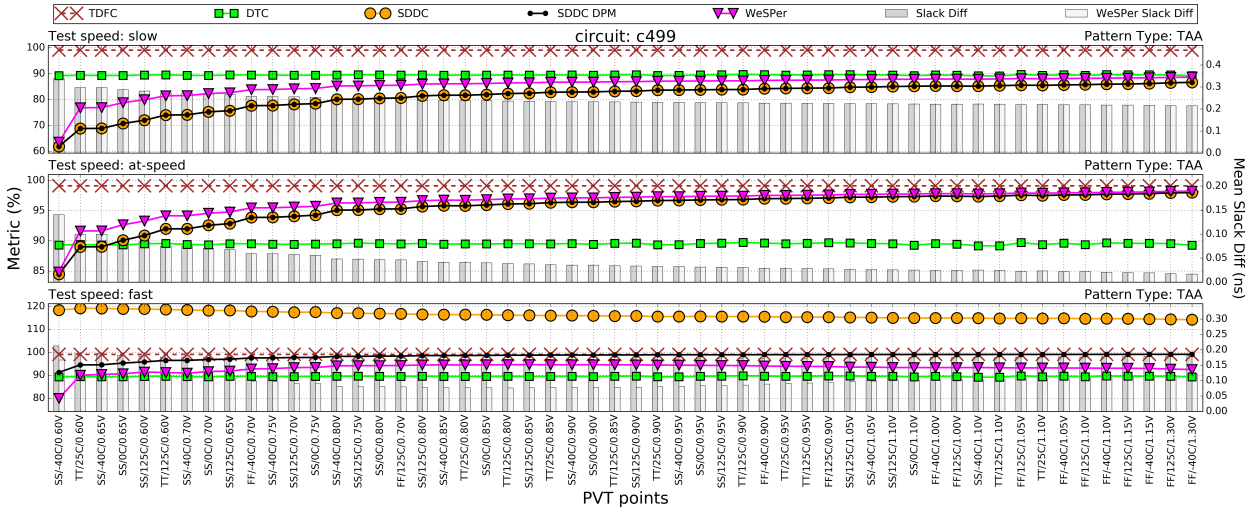

Figure A.15 Metric results for all available PVT points for the c499. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

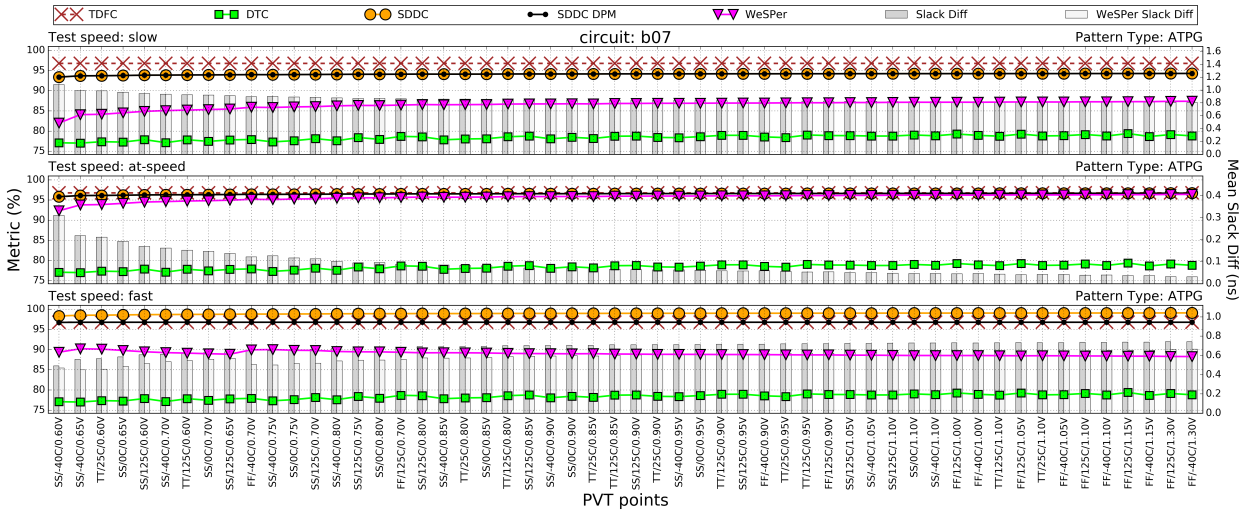

Figure A.16 Metric results for all available PVT points for the b07. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

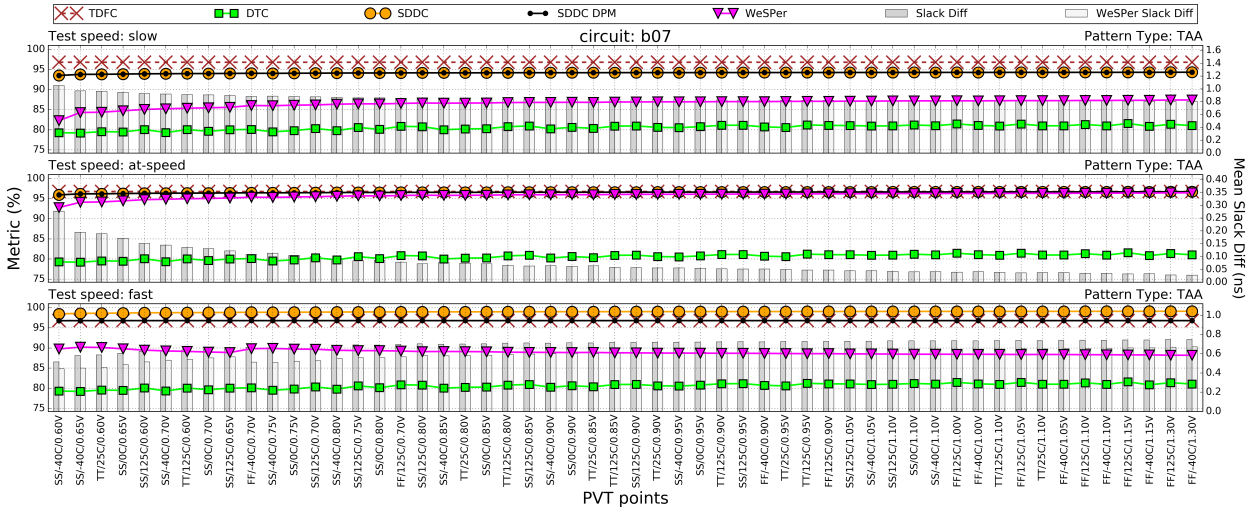

Figure A.17 Metric results for all available PVT points for the b07. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

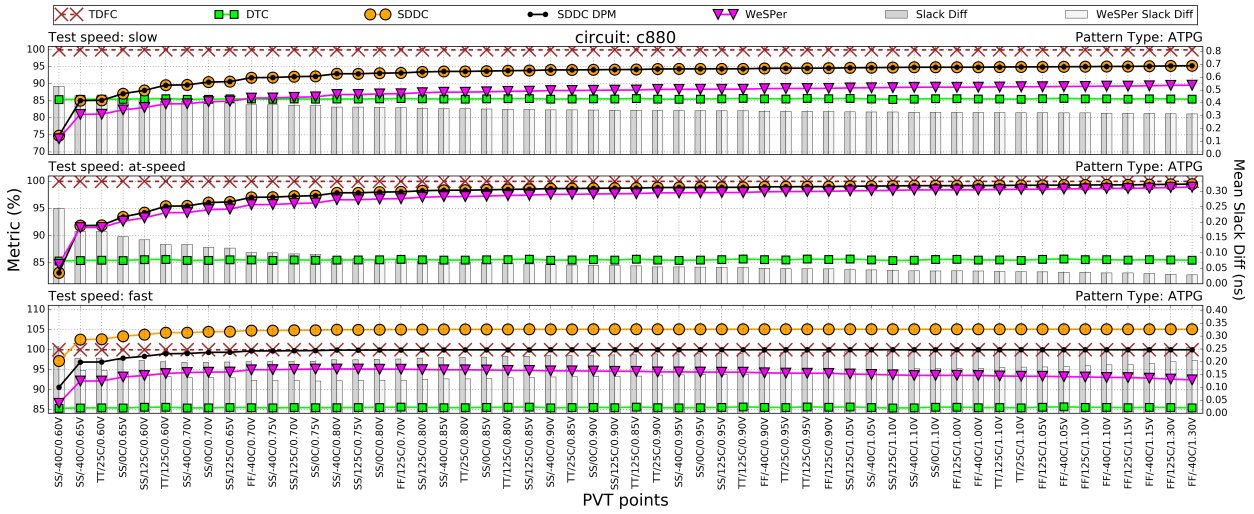

Figure A.18 Metric results for all available PVT points for the c880. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

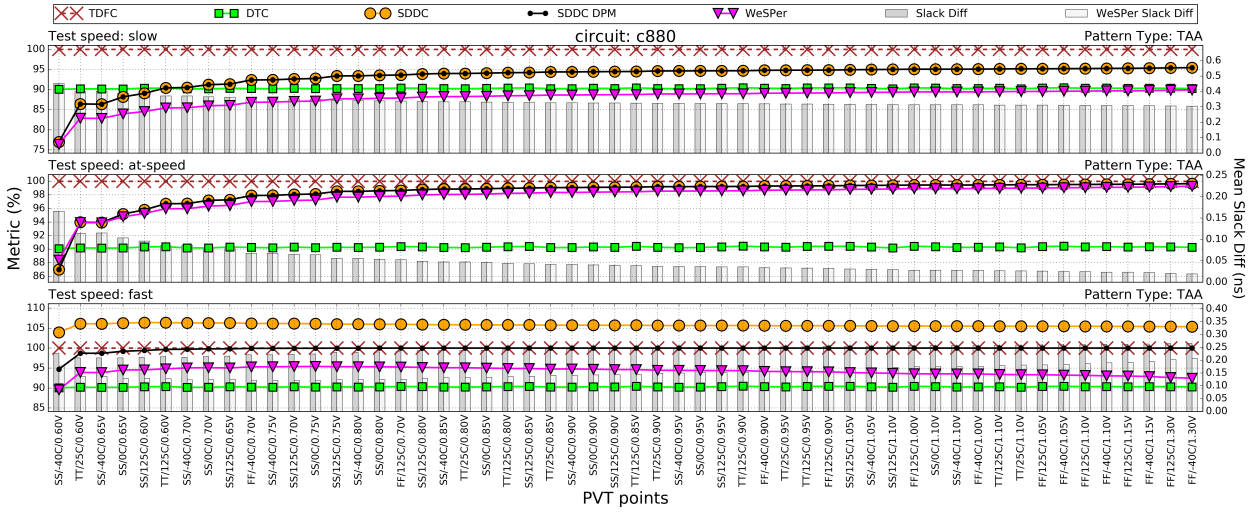

Figure A.19 Metric results for all available PVT points for the c880. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

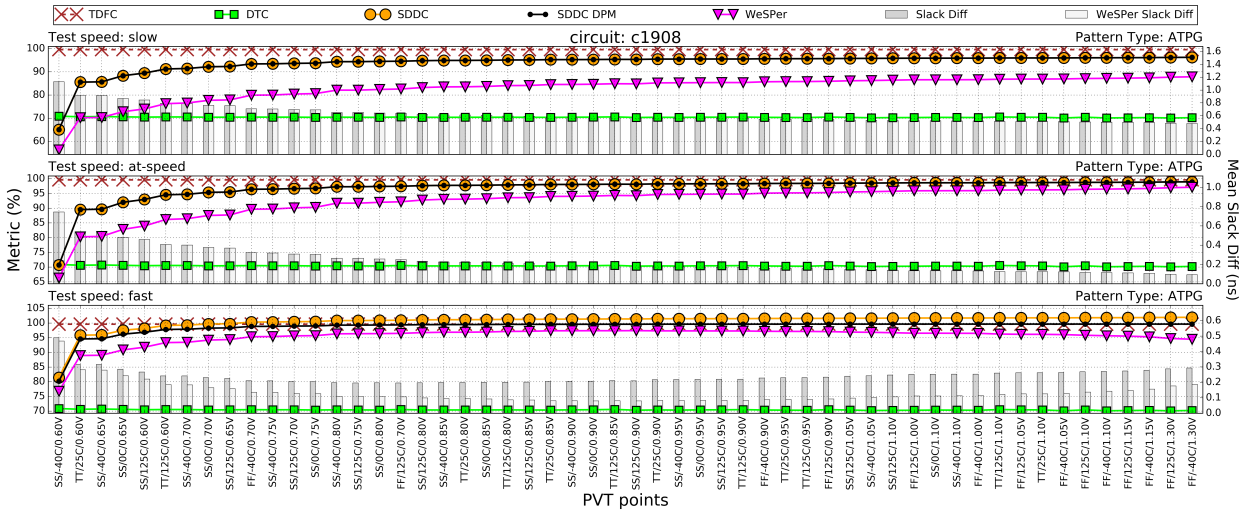

Figure A.20 Metric results for all available PVT points for the c1908. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

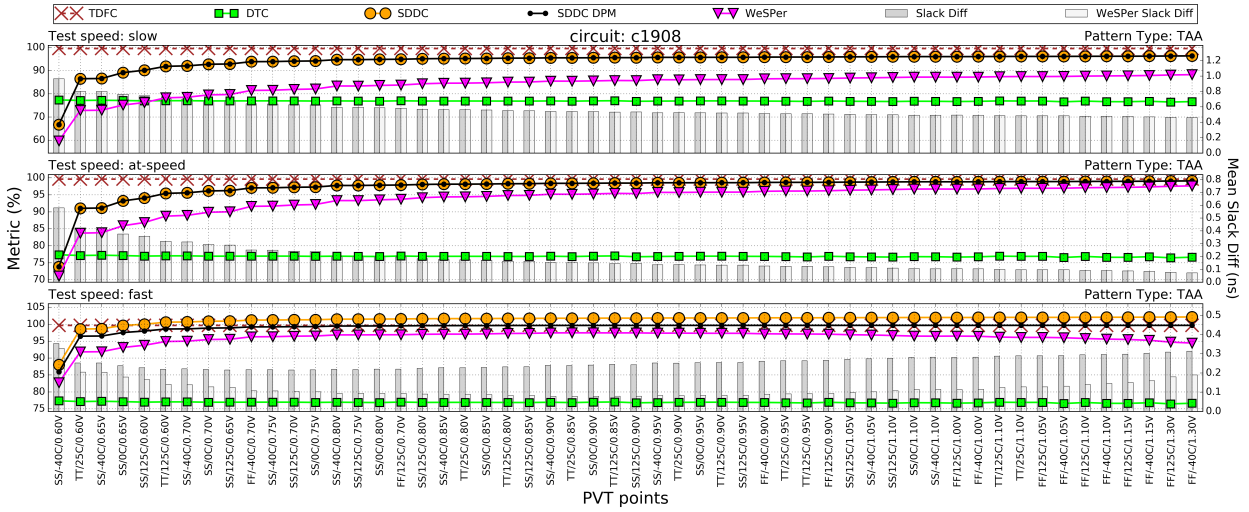

Figure A.21 Metric results for all available PVT points for the c1908. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

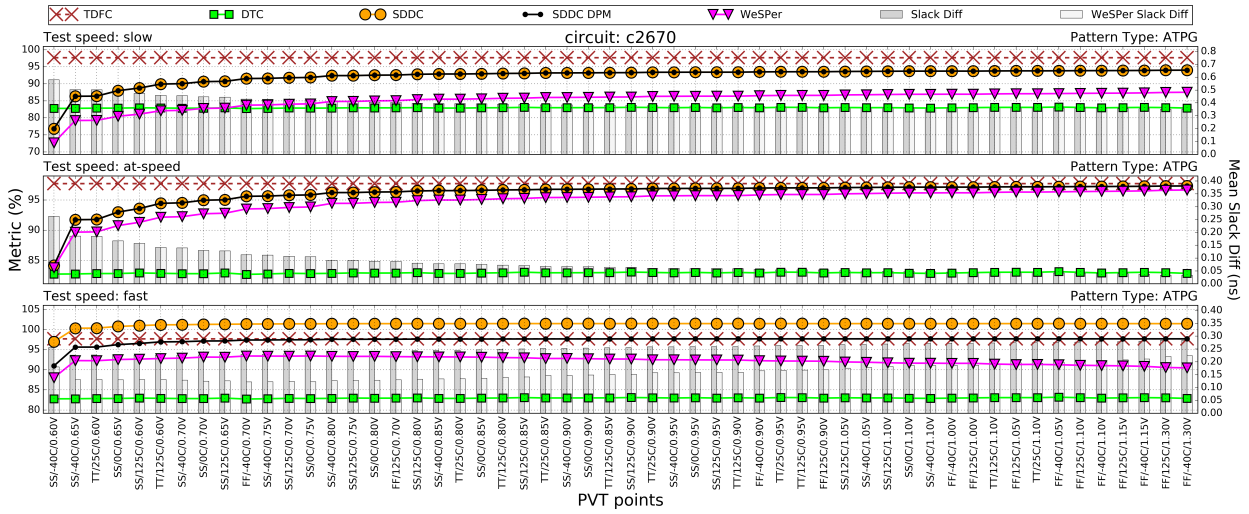

Figure A.22 Metric results for all available PVT points for the c2670. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

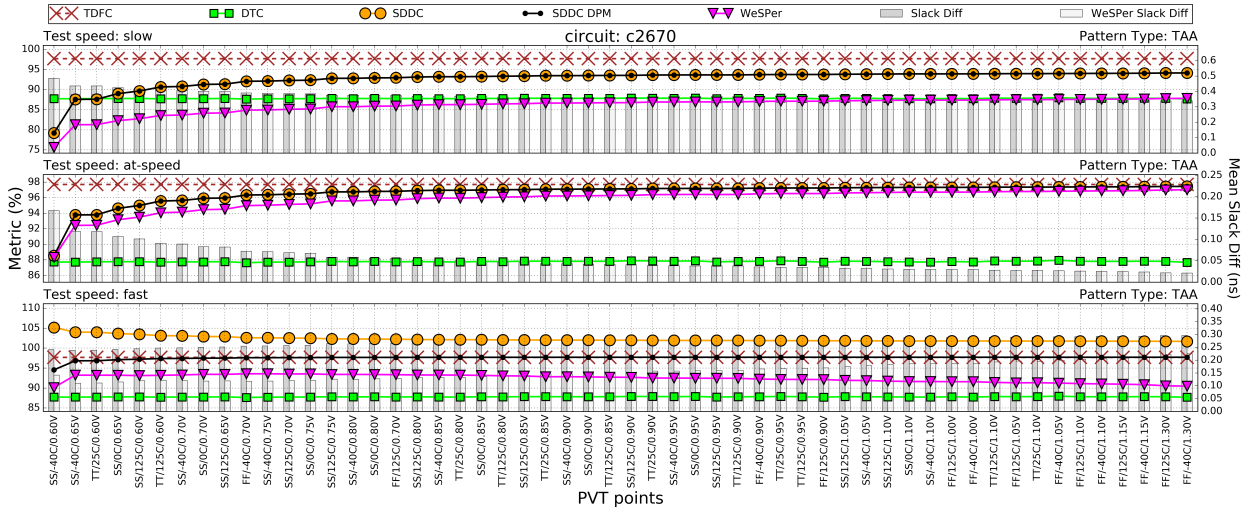

Figure A.23 Metric results for all available PVT points for the c2670. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

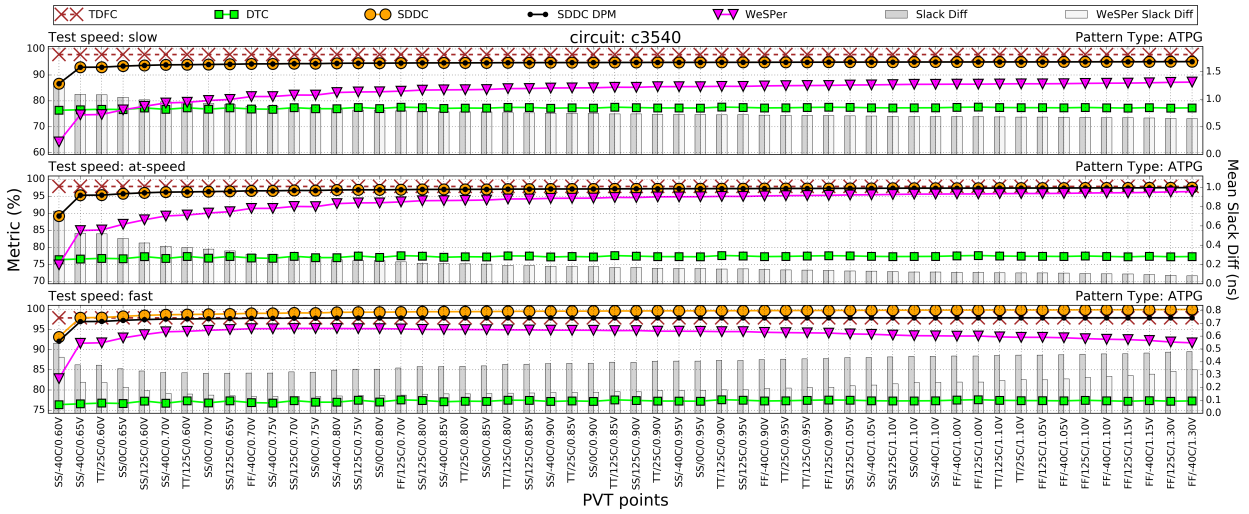

Figure A.24 Metric results for all available PVT points for the c3540. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

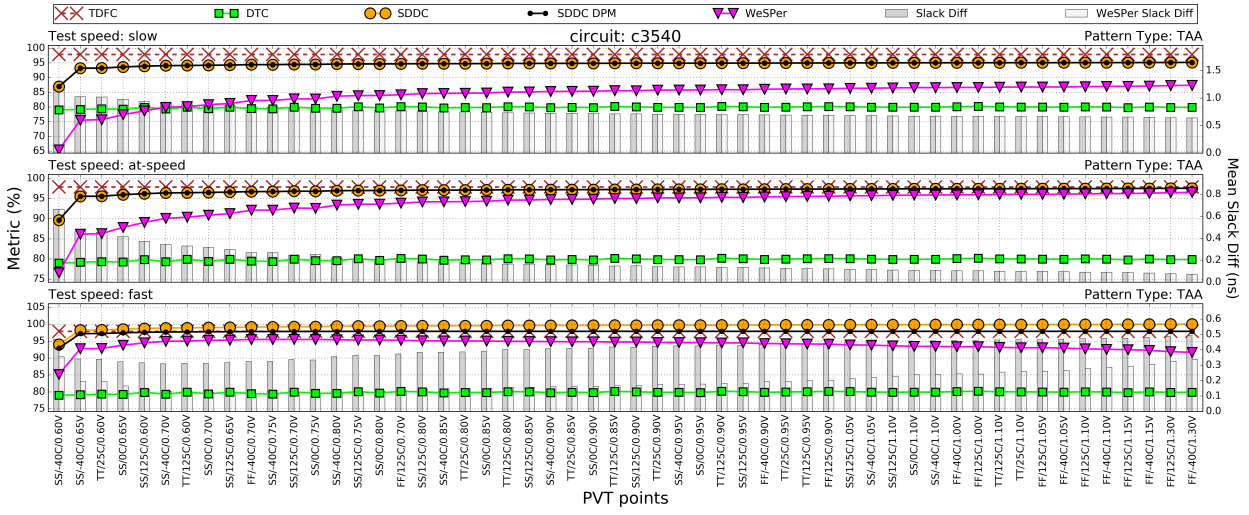

Figure A.25 Metric results for all available PVT points for the c3540. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

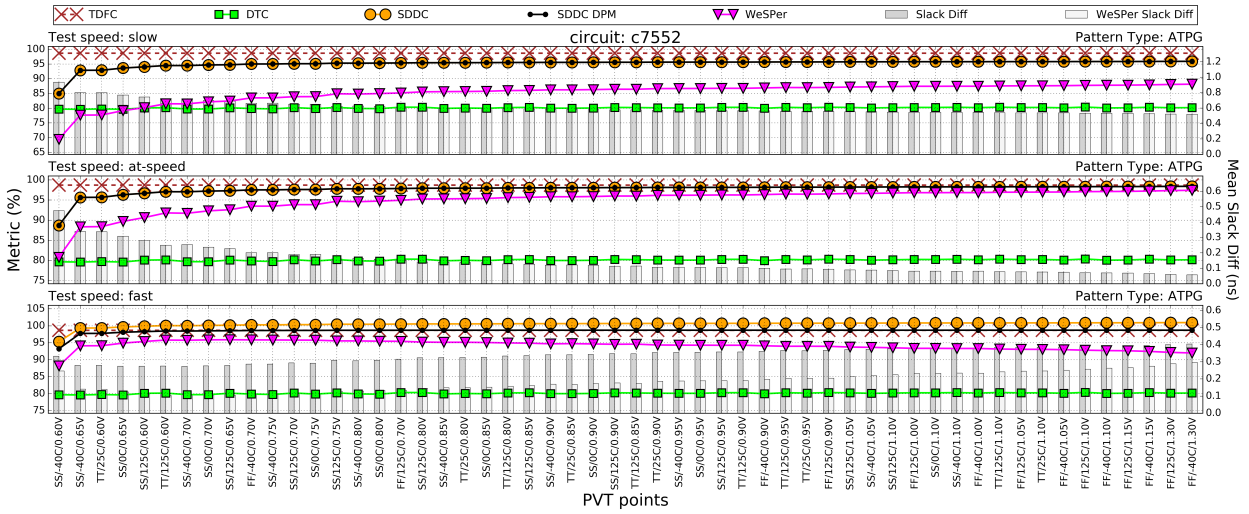

Figure A.26 Metric results for all available PVT points for the c7552. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular ATPG pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

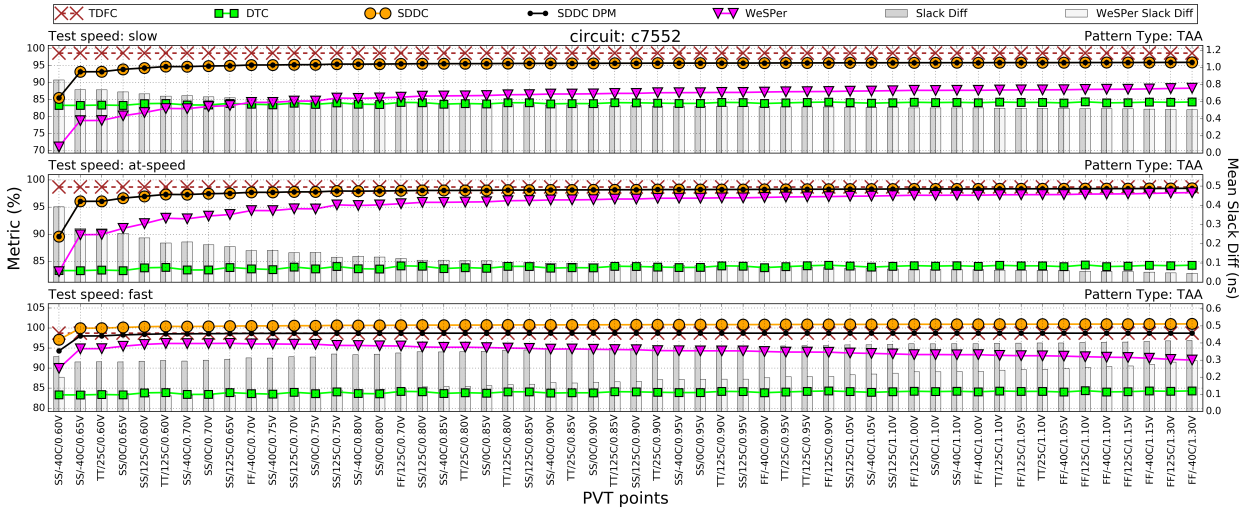

Figure A.27 Metric results for all available PVT points for the c7552. The left Y-axis shows metric values while the right Y-axis shows the mean slack difference values. These results were obtained with a regular TAA pattern set and are plotted for three test clock speeds: slow (top), at-speed (middle) and fast (bottom).

APPENDIX B TEST SETUP DETAILS

This appendix lists the test setup of the AnARM chips. This information is needed if the test has to be redone, whether on the same tester (Teradyne FLEX) or another.

Signals

The signals of the AnARM are categorized for testing into 4 groups. Inputs, outputs, power and ground. There is only one ground signal. The rest are divided into subgroups according to their purpose and timing needs. Table B.1 lists all the signals described in a test file or a test pattern. Notice that some of the signals have been used in the pattern file but they are not physically available on the chip or they do not matter for the test. Those signals are called dummy signals in Table B.1.

Patterns

There are 3 main pattern files that are included in the test sequence:

- Scan chain test
- Stuck-at test (labeled: DFT)
- At-Speed test (labeled: AST)

Some of those pattern files are divided into multiple files due to memory size limit or to inject faults on purpose. Table B.2 gives a summary of the size of the pattern and vectors in each of the listed pattern files. The total cycle number is the number of parallel or scan vector applications. The number of scan vectors in the total scan cycle is listed as the scan cycles. The parallel vector size is the number of bits (signals) defined in each pattern. The pattern length is the size of the pattern in term of parallel vectors (i.e. when applying scan vectors as parallel vectors). Note that in last application we assume that the scan chain are all padded with 0s and Xs in order to match the size of the longest scan chain (a padded pattern file is available for each pattern file). Lastly the pattern size in bit is calculated based on pattern length and parallel vector size (simple multiplication).

Table B.1 Grouping and directions of I/Os

Top Group	Sub Group	signals	Direction	Notes
DigitIN	GpIN	gpi___16___	input	
		gpi___15___	input	
		gpi___14___	input	
		gpi___13___	input	
		gpi___12___	input	
		gpi___11___	input	
		gpi___10___	input	
		gpi___9___	input	
		gpi___8___	input	
		gpi___7___	input	
		gpi___6___	input	
		gpi___5___	input	dummy, no physical connection
		gpi___4___	input	dummy, no physical connection
		gpi___3___	input	dummy, no physical connection
		gpi___2___	input	dummy, no physical connection
		gpi___1___	input	dummy, no physical connection
		gpi___0___	input	
	TestMode	testmode0	input	
		testmode1	input	
	FaultINJ	ast_fi___1___	input	fault injection
		ast_fi___0___	input	fault injection
	SCLK	clk_ref	input	scan clock
	JTAGin	DBGTDI	input	TDI
		DBGTMS	input	TMS
		DBGTRSTN	input	TRESTN
	JTAGclk	TCK	input	TCK
	RESET	NRESET	Input	reset signal negative
DigitOUT	GpOUT	gpo___15___	output	
		gpo___14___	output	
		gpo___13___	output	
		gpo___12___	output	
		gpo___11___	output	
		gpo___10___	output	
		gpo___9___	output	dummy
		gpo___8___	output	dummy
		gpo___7___	output	dummy
		gpo___6___	output	dummy
		gpo___5___	output	
		gpo___4___	output	
		gpo___3___	output	
		gpo___2___	output	
		gpo___1___	output	
		gpo___0___	output	
	JTAGout	DBGTDO	output	TDO
DummPow		gnd	I/O	dummy not used
		vdd	I/O	dummy not used
		gnde	I/O	dummy not used
		vdde	I/O	dummy not used
		vdd_oct	I/O	dummy not used

Table B.2 Patterns size summary

Pattern type	file name PAD_toplevel_c12t28soi_ ...	total cycle count	scan cycles	parallel vector size (bits)	pattern length (in parallel vectors)	pattern size (bits)
Scan Chain Test	iopad_pat	44	2	49	9636	472164
Stuck-at fault test part 1	dft1_pat___maxloads_200_01	607	202	49	969399	47500551
Stuck-at fault test part 2	dft1_pat___maxloads_200_02	604	201	49	964600	47265400
Stuck-at fault test part 3	dft1_pat___maxloads_200_03	604	201	49	964600	47265400
Stuck-at fault test part 4	dft1_pat___maxloads_200_04	604	201	49	964600	47265400
Stuck-at fault test part 5	dft1_pat___maxloads_200_05	604	201	49	964600	47265400
Stuck-at fault test part 6	dft1_pat___maxloads_200_06	604	201	49	964600	47265400
Stuck-at fault test part 7	dft1_pat___maxloads_200_07	604	201	49	964600	47265400
Stuck-at fault test part 8	dft1_pat___maxloads_200_08	604	201	49	964600	47265400
Stuck-at fault test part 9	dft1_pat___maxloads_200_09	604	201	49	964600	47265400
Stuck-at fault test part 10	dft1_pat___maxloads_200_10	604	201	49	964600	47265400
Stuck-at fault test part 11	dft1_pat___maxloads_200_11	604	201	49	964600	47265400
Stuck-at fault test part 12	dft1_pat___maxloads_200_12	604	201	49	964600	47265400
Stuck-at fault test part 13	dft1_pat___maxloads_200_13	604	201	49	964600	47265400
Stuck-at fault test part 14	dft1_pat___maxloads_200_14	604	201	49	964600	47265400
Stuck-at fault test part 15	dft1_pat___maxloads_200_15	157	52	49	249549	12227901
At-Speed Delay test	ast1_pat	818	205	49	983998	48215902

Time Plates

Time plates define the timing of when input signals are enabled within a frame of time (the test clock period) and when output signals are captured. The scan chain test has only one time plate that is used in its pattern file (gen_tp1 shown in Fig.B.1). Notice that in that figure the arrows sign that appears in the DigitOUT time plate represent the time when those signals are captured. In the same way the stuck-at fault has two time plates (gen_tp1 and gen_tp2) defined. However, they are identical and are shown in Fig B.2. There are three time plates for the at-speed test. The gen_tp1 is identical to the stuck-at fault test (compare Fig.B.2 and Fig.B.3). The other two time plates for the launch-on-capture are shown in Fig.B.4 and Fig.B.5.

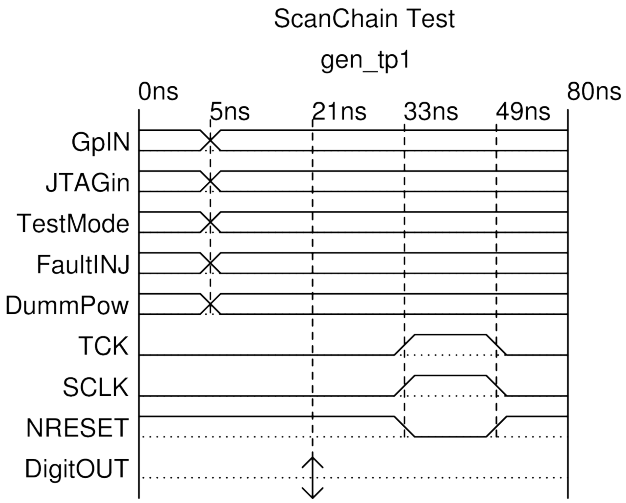

Figure B.1 The gen_tp1 time plate for the scan chain test.

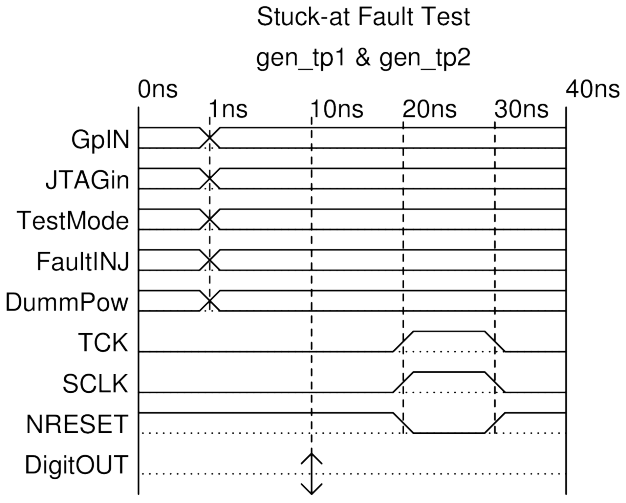

Figure B.2 The gen_tp1 time plate for the stuck-at fault test.

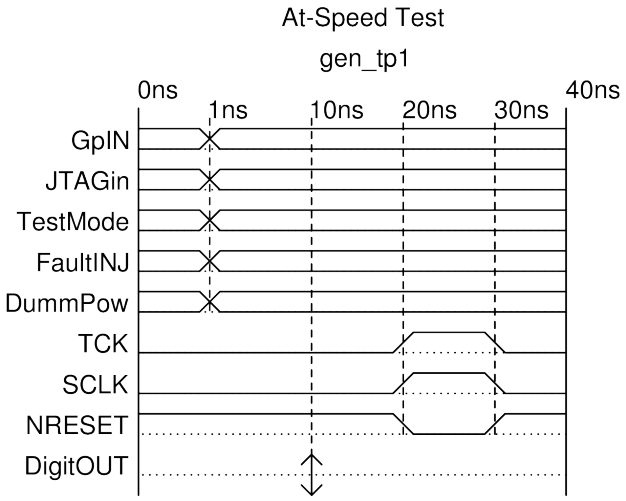

Figure B.3 The gen_tp1 time plate for the at-speed test.

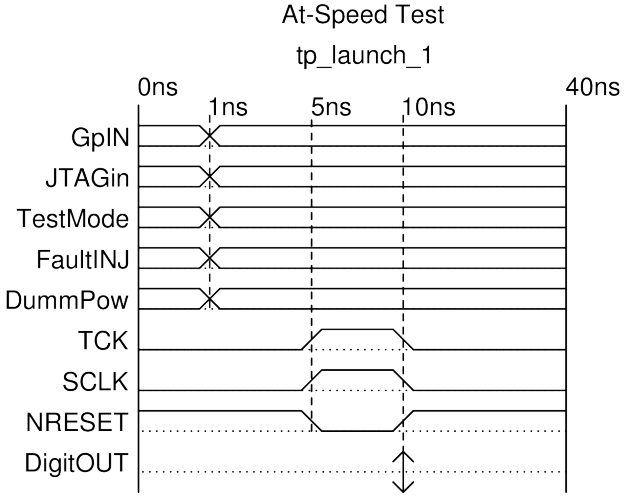

Figure B.4 The launch time plate for the at-speed test.

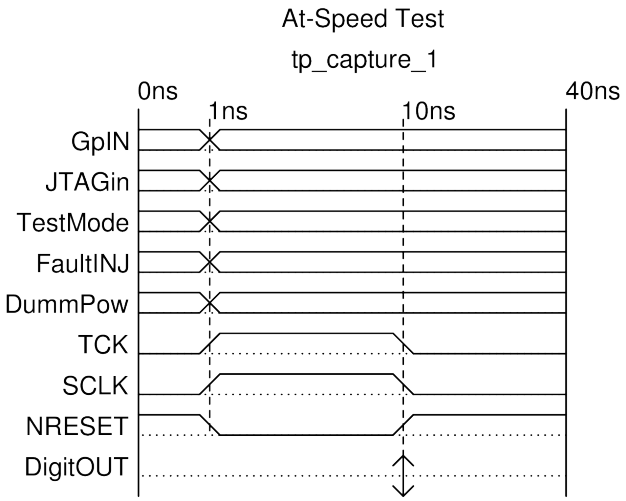

Figure B.5 The capture time plate for the at-speed test.